NAVAL RESEARCH LOGISTICS QUARTERLY

MARCH 1974
VOL. 21, NO. 1

OFFICE OF NAVAL RESEARCH

NAVSO P-1278

NAVAL RESEARCH LOGISTICS QUARTERLY

The Naval Research Logistics Quarterly is devoted to the dissemination of scientific information in logistics and will publish research and expository papers, including those in certain areas of mathematics, statistics, and economics relevant to the over-all effort to improve the efficiency and effectiveness of logistics operations.

Information for Contributors is indicated on inside back cover.

The Naval Research Logistics Quarterly is published by the Office of Naval Research in the months of March, June, September, and December and can be purchased from the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402. Subscription Price: $10.00 a year in the U.S. and Canada, $12.50 elsewhere. Cost of individual issues may be obtained from the Superintendent of Documents.

The views and opinions expressed in this quarterly are those of the authors and not necessarily those of the Office of Naval Research.

Issuance of this periodical approved in accordance with Department of the Navy Publications and Printing Regulations, NAVEXOS P-35

ASSEMBLY OF SYSTEMS HAVING MAXIMUM RELIABILITY*

Cyrus Derman

Columbia Univ.

Gerald J. Lieberman

Stanford Univ.

Sheldon M. Ross

Univ. of California at Berkeley

ABSTRACT

The first problem considered in this paper is concerned with the assembly of independent components into parallel systems so as to maximize the expected number of systems that perform satisfactorily. Associated with each component is a probability of it performing successfully. It is shown that an optimal assembly is obtained if the reliability of each assembled system can be made equal. If such equality is not attainable, then bounds are given so that the maximum expected number of systems that perform satisfactorily will lie within these stated bounds; the bounds being a function of an arbitrarily chosen assembly. An improvement algorithm is also presented.

A second problem treated is concerned with the optimal design of a system. Instead of assembling given units, there is an opportunity to "control" their quality, i.e., the manufacturer is able to fix the probability, p, of a unit performing successfully. However, his resources are limited so that a constraint is imposed on these probabilities. For (1) series systems, (2) parallel systems, and (3) k out of n systems, results are obtained for finding the optimal p's which maximize the reliability of a single system, and which maximize the expected number of systems that perform satisfactorily out of a total assembly of J systems.

[0] SUMMARY

The first problem considered in this paper is concerned with the assembly of independent components into parallel systems so as to maximize the expected number of systems that perform satisfactorily. Associated with each component is a probability of it performing successfully. It is shown that an optimal assembly is obtained if the reliability of each assembled system can be made equal. If such equality is not attainable, then bounds are given so that the maximum expected number of systems that perform satisfactorily will lie within these stated bounds; the bounds being a function of an arbitrarily chosen assembly. An improvement algorithm is also presented.

A second problem treated is concerned with the optimal design of a system. Instead of assembling given units, there is an opportunity to "control" their quality, i.e., the manufacturer is able to fix the probability, p, of a unit performing successfully. However, his resources are limited so that a constraint is imposed on these probabilities. For (1) series systems, (2) parallel systems, and (3) k out of n systems,

* This research was supported, in part, by the Office of Naval Research under contract N00014-67-A-0112-0052 with Stanford University and N00014-67-A-0108-0003 with Columbia University.

results are obtained for finding the optimal p's which maximize the reliability of a single system, and which maximize the expected number of systems that perform satisfactorily out of a total assembly of J systems.

1. INTRODUCTION

In a previous paper [1], the authors considered the following reliability problem: A system has n different types of components. Associated with each component is a numerical value. Let $\{a^m\}$ $(m=1, 2, \ldots, n)$ denote the set of numerical values of the n components. Let $R(a^1, a^2, \ldots, a^n)$ denote the probability that the system will perform satisfactorily, i.e., $R(a^1, a^2, \ldots, a^n)$ is the reliability of the system. Now suppose $a_1^m \leqslant a_2^m \leqslant \ldots \leqslant a_J^m$ are J components of type m $(m=1, 2, \ldots, n)$. Then J systems can be assembled from these components. Let N denote the number of systems that perform satisfactorily. N is a random variable whose distribution will depend on the way the J systems are assembled. The results obtained show that if $R(a^1, a^2, \ldots, a^n)$ has the properties of a joint cumulative distribution function then of all different ways in which the J systems can be assembled, $E(N)$ is maximized if these J systems have reliability $R(a_j^1, a_j^2, \ldots, a_j^n)$ $(j=1, 2, \ldots, J)$, i.e., assemble the best of each type, the next best of each type, . . ., and finally the worst of each type.

The aforementioned results are applicable to series systems of independent components, but are not applicable to parallel systems of independent components. Section 2 of this paper treats the problem of parallel systems of independent components, and shows that an optimal assembly is obtained if the reliability of each assembled system can be made equal. Furthermore, if equal reliability for each assembled system is not possible, then bounds are obtained so that the maximum expected number of systems that perform satisfactorily will lie within these stated bounds; the bounds being a function of an arbitrarily chosen assembly. An algorithm will be presented which will produce assemblies that result in an improvement in these bounds, although it will not necessarily lead to the optimal assembly. The results are not only applicable to the aforementioned problem, but are also applicable to the assembly of parallel systems of independent units, where the units in a system are interchangeable (only one type of unit is in the system). The results are also valid for the interchangeable and non-interchangeable cases when the number of units of a given type that must appear in a given system is not fixed in advance.

Section 3 is concerned with a variation of the assembly problem dealing with system design. Suppose a single system is to be constructed containing n units. Attached to the mth unit is a positive value p_m, which will denote the probability that the mth unit will perform satisfactorily, and these probabilities are assumed to be independent. The n units are to be "manufactured," and the manufacturer has sufficient control of his process that he is able to produce at an aimed at "p" level, but his resources are limited so that there is a constraint imposed on the values of the p's, namely $\sum_{m=1}^{n} p_m = A$. For (1) series systems, (2) parallel systems, and (3) k out of n systems (the system operates satisfactorily if at least k out of n units operate satisfactorily), the problem considered is to find the desired p's so as to maximize the reliability of the system. For a series system the optimal solution is to make all the p's equal to A/n. For the parallel system the optimal solution is to make one of the p's equal to A and the rest 0. For the k out of n system, the results are more complex.

Section 4 is concerned with the same problem considered in section 3, except that nJ components are to be "manufactured" and assembled into J systems, each being a k out of n system. Again, it is

desired to maximize the expected number of systems that perform satisfactorily, subject to constraints on the p's. A characterization of the optimal solution is given.

2. OPTIMAL ASSEMBLY OF PARALLEL SYSTEMS OF INDEPENDENT COMPONENTS

The following problem will be considered first. A set of M units are given which are numbered $1, 2, \ldots, M$. The M units are to be partitioned into J disjoint systems. After completion of a partition, the number of units contained in the jth system ($j=1, 2, \ldots, J$) will be denoted by n_j, with the added restriction that $\sum_{j=1}^{J} n_j = M$.* A system will perform satisfactorily if at least one of the n_j units in the system performs satisfactorily, i.e., it is a parallel system. Attached to the mth unit, $m=1, 2, \ldots, M$, is a positive value p_m which will denote the probability that the mth unit will perform satisfactorily, and these probabilities are assumed to be independent. For a given partition, the reliability of system j ($j=1, 2, \ldots, J$), R_j, is the probability that the system will perform satisfactorily, and can be expressed as†

$$R_j = 1 - \prod_{\substack{\text{all } m \text{ in} \\ \text{system } j}} (1-p_m). \tag{1}$$

Let N denote the number of systems that perform satisfactorily, so that N is a random variable whose distribution will depend upon a given partition. For a given partition, the expected number of systems that perform satisfactorily, $E(N)$, is then seen to be

$$E(N) = \sum_{j=1}^{J} \left[1 - \prod_{\substack{\text{all } m \text{ in} \\ \text{system } j}} (1-p_m)\right] = J - \sum_{j=1}^{J} \prod_{\substack{\text{all } m \text{ in} \\ \text{system } j}} (1-p_m). \tag{2}$$

The problem treated in this section is to find the partition which maximizes (2), or alternatively, to find the partition that minimizes the expression

$$\sum_{j=1}^{J} \prod_{\substack{\text{all } m \text{ in} \\ \text{system } j}} a_m, \tag{3}$$

where $a_m = (1-p_m)$. Henceforth, the a_m will be referred to as the positive value attached to the mth unit.

In a given partition, denote the set of n_j units appearing in system j ($j=1, 2, \ldots, J$) by A_j and define $|A_j|$ to equal the product of the values of the units in the set A_j. If A_j is the null set, $|A_j|$ is defined to be 1. Define

$$T = \text{Minimum} \sum_{j=1}^{J} |A_j|, \tag{4}$$

---

† A partition will allow for one or more systems to contain no units so long as $\sum_{j=1}^{J} n_j = M$. The reliability of a system containing no units will be taken to be zero.

where the minimum is taken over all possible partitions of the M units into J disjoint sets. The major result of this section shows that for any partition $A_1, A_2, \ldots, A_J$, T satisfies the following inequalities:

$$J \text{ Min } |A_j| \leq T \leq \sum_{j=1}^{J} |A_j|. \tag{5}$$

In order to prove this result, lemma 1 must be verified.

LEMMA 1: Let a_m and b_m $(m=1, 2, \ldots, s)$ be positive numbers.

If

$$\prod_{m=1}^{s} a_m = \prod_{m=1}^{s} b_m, \tag{6}$$

then

$$\sum_{m=1}^{s} (a_m/b_m) \geq s. \tag{7}$$

PROOF: It is a well known result that the arithmetic mean is always greater than or equal to the geometric mean, i.e.,

$$\sum_{m=1}^{s} y_m/s \geq \left(\prod_{m=1}^{s} y_m \right)^{1/s}. \tag{8}$$

Substituting $y_m = a_m/b_m$ into (8), it follows that

$$\sum_{m=1}^{s} (a_m/b_m) \geq s \left[\prod_{m=1}^{s} a_m / \prod_{m=1}^{s} b_m \right]^{1/s}. \tag{9}$$

From Equation (6), the right hand side of Inequality (9) becomes s, so that the lemma is proved.

THEOREM 1: For any partition $A_1, A_2, \ldots, A_J$, T satisfies the following inequalities

$$J \text{ Min } |A_j| \leq T \leq \sum_{j=1}^{J} |A_j|.$$

PROOF: Let $A_1, \ldots, A_J$ be any partition, and let $B_1, \ldots, B_J$ be the optimal partition. Using set theoretic identities, it follows that

$$|B_j| = \frac{|A_j|\,|B_j \bar{A}_j|}{|A_j \bar{B}_j|}, \qquad j=1, \ldots, J$$

where $\bar{A}_j$ and $\bar{B}_j$ are the complements of A_j and B_j, respectively. Summing over j, and since $B_1, \ldots, B_J$ is optimal

$$\begin{aligned} \sum_{j=1}^{J} |B_j| &= T \\ &\geq \min_j |A_j| \sum_{j=1}^{J} \frac{|B_j \bar{A}_j|}{|A_j \bar{B}_j|}. \end{aligned} \tag{10}$$

However, from Lemma 1, noting that

$$\prod_{j=1}^{J} |B_j\bar{A}_j| = \prod_{j=1}^{J} |A_j\bar{B}_j|,$$

we have that the second factor on the right of (10) is greater than or equal to J. Thus, the left inequality is proved. The right inequality follows from the definition of T. Hence, the theorem is proved.

The theorem indicates that the maximum expected number of systems that perform satisfactorily will lie within the stated bounds; these bounds being a function of the chosen partition. Furthermore, if a partition can be found that makes each system have the property that the product of the probabilities of each unit failing, i.e., the a's is equal, then this partition is optimal in that $\dot{E}(N)$ is maximized. These results lead to questioning whether or not an algorithm can be obtained which will determine the optimal partition. Unfortunately, the authors have been unable to find one, but an algorithm will be presented which should lead to a "good" solution; but not necessarily an optimal one.

A given partition results in a sequence of sets $A_1, A_2, \ldots, A_J$ (each set representing a system) and a corresponding $|A_1|, |A_2|, \ldots, |A_J|$. It can be assumed that the $|A|$'s are not equal; otherwise, an optimal partition has been obtained. Choose any two systems whose $|A|$'s are not equal and without loss of generality, denote them by A_1 and A_2, with $|A_2| > |A_1|$. It will be shown that under certain conditions units of one can be interchanged with units of the other, thereby resulting in a new partition with sharper bounds than given in (5). Let α_1 denote the product of the values which are attached to those units which are to be removed from A_1 and placed into A_2. Similarly, let α_2 denote the product of the values which are attached to those units which are to be removed from A_2 and placed into A_1. Thus, in the new partition, the $|A|$'s are given by $(\alpha_2/\alpha_1)\,|A_1|$, $(\alpha_1/\alpha_2)\,|A_2|$, $|A_3|, \ldots, |A_J|$. The main result to be obtained is that if $|A_2| > |A_1|$ and $||(\alpha_1/\alpha_2)|A_2| - (\alpha_2/\alpha_1)|A_1|\,|| < ||\,|A_2| - |A_1|\,||$,* then

$$|A_1| + |A_2| > (\alpha_2/\alpha_1)|A_1| + (\alpha_1/\alpha_2)|A_2|. \tag{11}$$

Before proving this result, two lemmas are required.

LEMMA 2: If $|A_2| > |A_1|$, then $|A_1| + |A_2| > (a_2/\alpha_1)|A_1| + (\alpha_1/\alpha_2)|A_2|$

$$\Leftrightarrow |A_1|/|A_2| < \alpha_1/\alpha_2 < 1.$$

PROOF: It is evident that $|A_1| + |A_2| > (\alpha_2/\alpha_1)|A_1| + (\alpha_1/\alpha_2)|A_2| \Leftrightarrow (\alpha_2 - \alpha_1)[|A_2|/\alpha_2 - |A_1|/\alpha_1] > 0$. If $|A_1|/|A_2| < \alpha_1/\alpha_2 < 1$, then $(\alpha_2 - \alpha_1)[|A_2|/\alpha_2 - |A_1|/\alpha_1] > 0$.
Now suppose that $(\alpha_2 - \alpha_1)[|A_2|/\alpha_2 - |A_1|/\alpha_1] > 0$. This implies that if $\alpha_2 - \alpha_1 < 0$, then $[|A_2|/\alpha_2 - |A_1|/\alpha_1] < 0$. However, this cannot hold because $\alpha_2 - \alpha_1 < 0$ and $|A_2| > |A_1|$ implies that $[|A_2|/\alpha_2 - |A_1|/\alpha_1] > 0$. Hence, $(\alpha_2 - \alpha_1) > 0$ and consequently, $[|A_2|/\alpha_2 - |A_1|/\alpha_1] > 0$. Therefore, $|A_1|/|A_2| < \alpha_1/\alpha_2 < 1$, and the Lemma is proved.

LEMMA 3: If $|A_2| > |A_1|$ and $|A_1| + |A_2| > (\alpha_2/\alpha_1)|A_1| + (\alpha_1/\alpha_2)|A_2|$, then $\|\,|A_2| - |A_1|\,\| > \|(\alpha_1/\alpha_2)|A_2| - (\alpha_2/\alpha_1)|A_1|\,\|$.

*The symbol $\|Z\|$ reads the absolute value of Z.

PROOF: Define the function $F(x)=x|A_2|-(1/x)|A_1|$, $0\leq x\leq 1$, and note that it is monotone increasing from $F(0)=-\infty$ to $F(1)=|A_2|-|A_1|$, with $F(|A_1|/|A_2|)=|A_1|-|A_2|$. Therefore, for all x such that $|A_1|/|A_2|<x<1$,

$$\|x|A_2|-(1/x)|A_1|\|<\||A_2|-|A_1|\|. \tag{12}$$

However, from Lemma 2, $|A_1|/|A_2|<\alpha_1/\alpha_2<1$. Thus, inequality (12) is satisfied for $x=\alpha_1/\alpha_2$, and the lemma is proved.

The converse of Lemma 3 is stated as Theorem 2.

THEOREM 2: If $|A_2|>|A_1|$ and $\|(\alpha_1/\alpha_2)|A_2|-(\alpha_2/\alpha_1)|A_1|\|<\||A_2|-|A_1|\|$, then

$$|A_1|+|A_2|>(\alpha_2/\alpha_1)|A_1|+(\alpha_1/\alpha_2)|A_2|.$$

PROOF: From the monotonicity of the function, F, defined in the proof of Lemma 3, and the values of $F(|A_1|/|A_2|)$ and $F(1)$ given in the proof of Lemma 3, it follows that (12) is satisfied for only those values of x such that $|A_1|/|A_2|<x<1$. It is then clear that $|A_1|/|A_2|<\alpha_1/\alpha_2<1$ is implied by the hypothesis of the theorem. The conclusion of the theorem follows from Lemma 2.

COROLLARY 1: For a given partition, if $|A_1|$ is the min $|A_j|$, $j=1, 2, \ldots, J$, and $|A_2|$ is any other, $|A_2|>|A_1|$, and α_1, α_2 are such that Theorem 2 holds, then min $\{(\alpha_2/\alpha_1)|A_1|, (\alpha_1/\alpha_2)|A_2|>|A_1|$. This implies that the new partition results in a higher lower bound than that given in (5).

PROOF: From the conclusion of Theorem 2, $|A_1|+|A_2|>(\alpha_2/\alpha_1)|A_1|+(\alpha_1/\alpha_2)|A_2|$. But from Lemma 2, this implies that

$$|A_1|/|A_2|<\alpha_1/\alpha_2<1.$$

Now, min $\{(\alpha_2/\alpha_1)|A_1|, (\alpha_1/\alpha_2)|A_2|\}>|A_1|$ if and only if $(\alpha_2/\alpha_1)|A_1|>|A_1|$ and $(\alpha_1/\alpha_2)|A_2|>|A_1|$. But these latter requirements are precisely the inequalities of Lemma 2, i.e., $|A_1|/|A_2|<\alpha_1/\alpha_2<1$.

Theorem 2 and Corollary 1 show that one iteration can sharpen both the upper and lower bounds given by (5). In fact, heuristically, one seeks partitions that tend to equalize the $|A|$'s, and this can be done systematically by interchanging units from one system with units from another system; these units satisfying the conditions of Theorem 2, e.g., interchanging units within the highest and lowest $|A|$'s. The algorithm would be continued until there are no pairwise interchanges satisfying the conditions of Theorem 2. When this occurs, the solution is "good" but not necessarily optimal. This can be seen by examining the following counterexample. Suppose there are nine units with associated a's to be divided into three systems as follows:

System 1:	0.09,	0.05,	0.05;	$\|A_1\|=0.000225$
System 2:	0.03,	0.24,	0.02;	$\|A_2\|=0.000144$
System 3:	0.12,	0.10,	0.015;	$\|A_3\|=0.000180$

For this partition $|A_1|+|A_2|+|A_3|=0.000549$, and cannot be improved by pairwise interchanges. However, consider the following partition:

System 1:	0.12,	0.05,	0.03;	$\|A_1\|=0.000180$
System 2:	0.05,	0.24,	0.015;	$\|A_2\|=0.000180$
System 3:	0.09,	0.10,	0.02;	$\|A_3\|=0.000180$

For this partition, which is, in fact, optimal, $|A_1|+|A_2|+|A_3|=0.000540$. Since the aforementioned algorithm only allows for pairwise interchanges, it need not lead to optimal solutions.

The problem considered to date has been in the context of taking M units and partitioning them into J disjoint systems, with units being interchangeable and the number required for each system not specified. Suppose the problem is now changed so that each system contains n different (non-interchangeable) types of components and J systems have to be assembled from the $M=nJ$ units, i.e., there are J units of each type available. Again, each system operates as a parallel system, and as before the objective is to find the partition that maximizes the expected number of systems that perform satisfactorily. How does this new problem compare with the problem previously treated? Fortunately, the admissible partitions for this new problem is a subset of the partitions of the original problem, and furthermore, none of the results depended on the number of units assigned to each system. Hence, all the results previously obtained are applicable to the new problem. Of course, similar comments can be made for the case of interchangeable components, but where each system must contain $n=M/J$ units.

3. SINGLE SYSTEM PROBLEM

Another variation of the assembly problem is concerned with system design. Suppose a single system is to be constructed containing n units. Three cases will be considered, namely (1) the system will perform satisfactorily if all of the n units performs satisfactorily, i.e., it is a series system; (2) the system will perform satisfactorily if at least one of the n units performs satisfactorily, i.e., it is a parallel system; and (3) the system will perform satisfactorily if at least $k>1$ units performs satisfactorily, i.e., it is a k out of n system. Attached to the mth unit is a value p_m, $m=1, 2, \ldots, n$, which will denote the probability that the mth unit will perform satisfactorily, $0 \leq p_m \leq 1$, and these probabilities are assumed to be independent. The n units are to be "manufactured," and the manufacturer has sufficient control of his process that he is able to produce at an aimed at "p" level, but his resources are limited so that there is a constraint imposed on the values of the p's, namely $\sum_{m=1}^{n} p_m=A$, where A is a fixed positive number. The problem is to find the desired p's so as to maximize the reliability of the system.

CASE 1: Series System

For a series system, the problem is to find the $p_1, p_2, \ldots, p_n$ which maximizes the reliability, R, where

$$R=\prod_{m=1}^{n} p_m,$$

subject to

$$0 \leq p_m \leq 1,\ m=1, 2, \ldots, n \text{ and } \sum_{m=1}^{n} p_m=A.$$

It can be assumed that $A \leq n$, otherwise the solution is to choose each $p_m=1$. The problem is equivalent to maximizing $\sum_{m=1}^{n} \log p_m$ subject to the same conditions. Since $\log p_m$ is a concave function and ignoring the constraints $p_m \leq 1$, $m=1, 2, \ldots, n$, it is well known that the optimal values of $p_1, p_2, \ldots, p_n$ are $p_1^*=p_2^*=\ldots=p_n^*=A/n$. Since $A \leq n$, the ignored constraints are satisfied. Hence $p_1^*=p_2^*=\ldots=p_n^*=A/n$ is optimal.

CASE 2: Parallel System

For a parallel system, the problem is to find the $p_1, p_2, \ldots, p_n$ which maximizes the reliability R, where

$$R=1-\prod_{m=1}^{n}(1-p_m),$$

subject to

$$0 \leqslant p_m \leqslant 1,\ m=1, 2, \ldots, n \text{ and } \sum_{m=1}^{n} p_m=A.$$

It can be assumed that $A<1$, otherwise the optimal solution is to choose at least one of the p's equal to 1 and the rest arbitrary (but subject to the constraint that $\sum_{m=1}^{n} p_m=A$). The problem is equivalent to minimizing

$$\sum_{m=1}^{n} \log (1-p_m),$$

subject to

$$p_m \geqslant 0,\ m=1, 2, \ldots, n, \text{ and } \sum_{m=1}^{n} p_m=A.$$

The constraint $p_m \leqslant 1$, $m=1, 2, \ldots, n$ is superfluous since $A<1$. The functions $\log (1-p_m)$ are concave. Therefore $\sum_{m=1}^{n} \log (1-p_m)$ is a concave function in $(p_1, p_2, \ldots, p_n)$ with the minimum at an extreme point of $\sum_{m=1}^{n} p_m=A$, $p_m \geqslant 0$ $(m=1, 2, \ldots, n)$. Each extreme point has one $p_m=A$ and the remaining equal to zero. Thus, in particular, $p_1^*=A$, $p_2^*=p_3^*=\ldots=p_n^*=0$ is optimal.

CASE 3: k out of n System

For a k out of n system, the system will perform satisfactorily if at least k out of n units perform satisfactorily. Let X_m, $m=1, 2, \ldots, n$, be independent Bernoulli random variables with parameter p_m. Let $Y=X_1+X_2+\ldots+X_n$. Then the reliability R can be expressed as

$$R=P\{Y \geqslant k\}.$$

The problem is to maximize R subject to

$$0 \leqslant p_m \leqslant 1, \text{ and } \sum_{m=1}^{n} p_m \leqslant A.$$

It can be assumed that $A<k$, otherwise the optimal solution is to choose k of the p's equal to 1 and the rest arbitrary. If the constraints $p_m \leqslant 1$ are ignored, the Kuhn-Tucker conditions indicate that if $p_1^*, p_2^*, \ldots, p_n^*$ are optimal, then there exists a number $\lambda^* \geqslant 0$ satisfying the following:

$$\text{If } p_m^*=0, \text{ then } \frac{\partial}{\partial p_m} P\{Y \geqslant k\}-\lambda^* \leqslant 0 \text{ at } p_m=p_m^*,$$
$$\text{for } m=1, 2, \ldots, n,$$

If $p_m^* > 0$, then $\frac{\partial}{\partial p_m} P\{Y \geq k\} - \lambda^* = 0$ at $p_m = p_m^*$,

for $m = 1, 2, \ldots, n$,

If $\lambda^* > 0$, then $\sum_{m=1}^{n} p_m^* = A$.

Suppose there are r variables which are greater than zero in the optimal solution and denote them by $p_1^*, p_2^*, \ldots, p_r^*$. The $P\{Y \geq k\}$ can be expressed as

$$P\{Y \geq k\} = p_1 F(p_2, p_3, \ldots, p_n) + (1 - p_1) G(p_2, p_3, \ldots, p_n),$$

where the function F is $P\{X_2 + X_3 + \ldots + X_n \geq k - 1\}$ and the function G is $P\{X_2 + X_3 + \ldots + X_n \geq k\}$. Hence,

$$\frac{\partial}{\partial p_1} P\{Y \geq k\} = F(p_2, p_3, \ldots, p_n) - G(p_2, p_3, \ldots, p_n) = H(p_2, p_3, \ldots, p_n).$$

Therefore, $H(p_2^*, p_3^*, \ldots, p_n^*) = \lambda^*$ can be solved uniquely for p_2^* as a function of $p_3^*, p_4^*, \ldots, p_n^*$ and λ^*.
Similarly,

$$\frac{\partial}{\partial p_2} P\{Y \geq k\} = H(p_1, p_3, \ldots, p_n) = \lambda^*$$

can be solved uniquely for p_1^* as the same function of $p_3^*, p_4^*, \ldots, p_n^*$ and λ^* sc that $p_1^* = p_2^*$. In the same manner, it can be shown that $p_1^* = p_2^* = \ldots = p_r^*$, and furthermore, it is clear that $p_i^* = A/r$, $i = 1, 2, \ldots, r$. If $\frac{A}{r} \leq 1$, then the ignored constraints are satisfied. This is easily shown to be the case. It is clear that $r \geq k$ since if more than $(n - k)$ p's are zero, then $P\{Y \geq k\} = 0$, which cannot be a maximum. Therefore $\frac{A}{r} \leq \frac{A}{k} < 1$, so that the ignored constraints are satisfied. The foregoing results do not indicate how to determine r. One way is to evaluate $P\{Y \geq k\}$ with $p_i^* = A/r$, $i = 1, 2, \ldots, r$, choosing that value of $r = r^*$ which maximizes

$$P\{Y \geq k\} = \sum_{j=k}^{r} \binom{r}{j} \left(\frac{A}{r}\right)^j \left(1 - \frac{A}{r}\right)^{r-j} \qquad \text{for } r = k, k+1, \ldots, n. \tag{13}$$

Some insight is obtained by replacing the right hand side of (13) by the normal or Poisson approximation, depending upon which is appropriate. When r is large and A/r is near 0 or 1 the Poisson is appropriate; when A/r is "near" $\frac{1}{2}$, the normal is appropriate. For either case, k (and consequently r) and n should be large. For the normal approximation $r = r^*$ is to be chosen which maximizes

$$\int_{\frac{k - r(A/r)}{\sqrt{r(A/r)(1 - A/r)}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz = \int_{\frac{k - A}{\sqrt{A(1 - A/r)}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz, \tag{14}$$

for $r = k, k+1, \ldots, n$. Since $k - A > 0$, clearly $r^* = n$. Thus,

$$p_1^* = p_2^* = \ldots = p_n^* = A/n. \tag{15}$$

If the Poisson approximation is used for the smaller values of A, the right hand side of (13) yields

$$P\{Y \geq k\} = 1 - \sum_{j=0}^{k-1} \frac{e^{-A}A^j}{j!},$$

which is independent of r. That is, approximately speaking, the choice of r has very little influence on the left hand side of (13). Therefore, it can be said that if n and k are both large, an approximately optimal solution is given by (15).

The results given in (15) are consistent with the n of n case (series system). It does not appear to be consistent with the 1 of n case (parallel system). However, this is not unexpected since k must be large in order for the approximation to be good.

The solution given in (15) is easily shown to be exact and optimal if the problem considered is to maximize the variance of the random variable Y, the number of units that perform satisfactorily, subject to the usual constraints. (Since $A < k$ and A is the expected number of components that function satisfactorily, maximization of the variance is desired.)

4. MULTIPLE SYSTEM PROBLEM

The problem considered in this section is the same as that considered in section 3, except that nJ components are to be "manufactured" and assembled into J systems. Each system performs satisfactorily if at least k out of n units perform satisfactorily. It is desired to maximize the expected number of systems that perform satisfactorily, $E(N)$, subject to constraints on the p's.

Motivated by the results obtained in section 3 using the approximation, it will be assumed initially that the probability of each unit performing satisfactorily within the system will be equal, i.e., all units in the jth system have probability p_j of functioning. The problem is to determine $p_1, p_2, \ldots, p_J$ to maximize

$$D(p_1, p_2, \ldots, p_J) = \sum_{j=1}^{J} \sum_{i=k}^{n} \binom{n}{i} p_j^i (1-p_j)^{n-i}, \tag{16}$$

subject to

$$0 \leq p_j \leq 1, \qquad j = 1, 2, \ldots, J$$

and

$$\sum_{j=1}^{J} np_j \leq A, \qquad \text{where } 0 \leq A < nJ.$$

The Kuhn-Tucker conditions indicate that if $p_1^*, p_2^*, \ldots, p_J^*$ are optimal, then there must exist numbers $\lambda^* \geq 0$, $\mu_j^* \geq 0$, $j = 1, 2, \ldots, n$, satisfying the following:
If

$$p_j^* = 0, \text{ then } \frac{\partial}{\partial p_j} D(p_1, p_2, \ldots, p_J) - n\lambda^* - \mu_j^* \leq 0 \text{ at } p_j = p_j^*, \qquad \text{for } j = 1, 2, \ldots, J. \tag{17a}$$

If

(17b) $p_j^* > 0$, then $\frac{\partial}{\partial p_j} D(p_1, p_2, \ldots, p_J) - n\lambda^* - \mu_j^* = 0$ at $p_j = p_j^*$, for $j = 1, 2, \ldots, J$.

If

(17c) $\mu_j^* = 0$, then $p_j^* \leq 1$ for $j = 1, 2, \ldots, J$.

If

(17d) $\mu_j^* > 0$, then $p^* = 1$ for $j = 1, 2, \ldots, J$.

If

(17e) $$\lambda^* = 0, \text{ then } n \sum_{j=1}^{J} p_j^* \leq A.$$

If

(17f) $$\lambda^* > 0, \quad \text{then } n \sum_{j=1}^{J} p_j^* = A.$$

Now,

$$\frac{\partial D}{\partial p_j} = n \binom{n-1}{n-k} p_j^{k-1} (1-p_j)^{n-k},$$

so that $\frac{\partial D}{\partial p_j} \geq 0$ for $0 \leq p_j \leq 1$ and equal to zero when $p_j = 0$ or 1. If $\mu_j^* > 0$, then $p_j^* = 1$. When $p_j^* = 1$, $\frac{\partial D}{\partial p_j} = 0$ so that $\mu_j^* = -n\lambda^*$ from (17b); this is a contradiction, and hence, $\mu_j^* = 0$ for all $j, j = 1, 2, \ldots, J$. Now if any $p_j^* = 1$, then from (17b) $\lambda^* = 0$, so that every p_j^* must be zero or one (from 17a or 17b) since $\frac{\partial D}{\partial p_j} = 0$ only at these values of p. If $0 < p_j < 1$, then from (17b)

$$n \binom{n-1}{n-k} p_j^* (1-p_j^*)^{n-k} = n\lambda^* > 0.$$

For a given λ^* the equation has at most two possible roots in the range $0 < p_j < 1$. Therefore, an optimal solution is of the form $p_1^* = p_2^* = \ldots = p_x^* = 0$, and $p_{x+1}^* = p_{x+2}^* = \ldots = p_{x+y} = \bar{p}, p_{x+y+1}, p_{x+y+2}, \ldots, p_J = \bar{\bar{p}}$, for some x and y, where $\bar{p}, \bar{\bar{p}}$ are solutions to

(18a) $$n \binom{n-1}{n-k} p^{k-1} (1-p)^{n-k} = \lambda^*, \quad \text{for some } \lambda^* > 0$$

and

(18b) $$y\bar{p} + (J - x - y)\bar{\bar{p}} = \frac{A}{n}.$$

An optimal solution may consist of $[s]p$'s each having value one and $(J-[s])p$'s each having value zero, where $[s]$ is the greatest integer less than or equal to A/n. But this is consistent with the given form when $y=0$, $\bar{p}=0$, $\bar{\bar{p}}=1$.

The curve $\sum_{i=k}^{n}\binom{n}{i}p^i(1-p)^{n-i}$ being s shaped is concave over the range $\frac{k-1}{n-1}\leq p\leq 1$. It is evident that if A/nJ satisfies $\frac{k-1}{n-1}\leq\frac{A}{nJ}\leq 1$, then $p_1^*=p_2^*=\ldots=p_J^*=\frac{A}{nJ}$ is optimal.

When $k=n$, the series case, the optimal solution is to have $p_j=1$ for $j=1,2,\ldots,[A/n]$, $p_{[A/n]+1}=A/n-[A/n]$, and $p_j=0$ for $j=[A/n]+2,\ldots,J$. This is immediate from convexity arguments, independent of the foregoing argument.

At the outset of this section it was assumed that the probability of each unit performing satisfactorily within a system was equal. This may not always be reasonable. In section 3, it was evident that when $A\geq k$, the optimal solution was to choose k of the p's equal to 1 and the rest 0. In the problem encountered in this section where nJ units are to be assembled into J systems, it may very well be that allocating k of the total resources to a system is appropriate even if $A\leq kJ$ and hence, the initial assumption about equal probabilities within a system may not be valid. Thus, any algorithm which seeks an optimal solution to this problem requires a consideration of such possible systems, and their contribution to the total of the expected number of systems that perform satisfactorily as well as allocating the remaining resources so as to maximize D in Equation (16).

REFERENCES

[1] Derman, C., Lieberman, G. J., and Ross, S. M., *On Optimal Assembly of Systems*, Nav. Res. Log. Quart. *19*, 569–574(1972).

A PARTITIONING TECHNIQUE FOR OBTAINING SOLUTIONS TO THE MODULARIZATION PROBLEM*

A. J. Caponecchi†

Air Force Systems Command
Wright-Patterson AFB, Ohio

and

P. A. Jensen

Department of Mechanical Engineering
The University of Texas at Austin

ABSTRACT

A significant problem in electronic system design is that of partitioning the functional elements of an equipment schematic into subsets which may be regarded as modules. The collection of all such subsets generated by a particular partitioning forms a potential modular design. The specific problem is to determine that partitioning of the schematic that minimizes a cost function defined on the subsets subject to specified hardware, design, packaging, and inventory constraints. This problem is termed the modularization problem. This paper presents a method for obtaining restricted solutions to the modularization problem by employing some recent developments in linear graph theory obtained by one of the coauthors. Numerical results from the solution of several typical problems are presented.

I INTRODUCTION

Life Cycle and Relative Support Costs

During the post World War II time period it has become increasingly apparent to the military community that the cost of acquiring new equipments to satisfy continuously changing requirements represents only one of the major cost factors associated with hardware ownership. Another major factor is the cost of equipment support. The costs of maintenance and logistics support are two categories which make up a significant portion of these follow-on demands for defense dollar resources over the planned life of the equipment.

It has been estimated by Goldman [9] that 20 to 30 percent of the total defense dollar goes to the support of existing equipments during the operational phase. Goldman further asserts that life time support costs often dominate system life cycle costs by an order of at least 10 times the original cost of the end unit.

Other investigators have placed different estimates on the cost of support. For example, Thomas D. Morris, a past Assistant Secretary of Defense for Installations and Logistics estimates [16] the cost for logistics support at approximately 70 percent of the Defense Budget. Still other estimates place support costs in a range of 10 to 100 times the initial procurement costs.

---

* Presented at the 40th National ORSA Meeting, October 27–29, 1971, Anaheim, California, The views expressed herein are those of the authors and do not necessarily reflect those of the United States Air Force or the Department of Defense.

† Major, USAF.

It is apparent that the contribution of support costs to life cycle costs depends on the specific system under consideration. The important fact is that the costs required to keep a modern system operational can represent a significant demand on the DOD Budget.

There are a large number of identifiable costs that can be classed as system support costs. Among these are the cost of operators, management, maintenance personnel, facilities, spares, and test equipment maintenance. Some of these costs are sentitive to decisions concerning the engineering design of a system and some are not. Goldman [9] points out that a principal determinant of support costs are the support system management decisions made throughout the life cycle. These can overshadow most individual design decisions. He further points out however that design decisions between alternatives having the same performance capability can be handled on the basis of relative support costs.

This paper is concerned with engineering design decisions and thus only those costs that are affected by these decisions will be considered. The techniques introduced here will be applicable during the engineering design phase of an equipment. During this phase it is the relative support costs of alternative designs that are the primary issue.

Modularized Designs

Modular designs are characterized by "plug-in" unitized construction. Selected groups of system components are placed in a single "package" which is readily removed from the system in the event of failure of one or more components contained within the package. Such packages are referred to as modules.

A given replaceable package may itself be made up of smaller replaceable modules. A modular design consisting of packages within which there exists a further modular breakdown is referred to as a multilevel modularized design. In this paper only single level modularized designs are considered.

The increased equipment complexities of modern electronic equipment have resulted in tremendous increases in equipment part counts. This growth in part count tends to offset the significant improvements which have been achieved in recent years in component reliabilities. In this environment, equipment downtimes for maintenance become a serious design consideration. The functional grouping of the system components into modules permits reduction of equipment downtimes by permitting rapid location and replacement of the failed item. Modularized construction can result in increased equipment availability while reducing maintenance support requirements.

Given the complete functional design of a system (schematics, blueprints, etc.) there exists a very large number of permissible partitions of the complete system into modular units. Any given partition has a distinct impact on the requirements which will be necessary for support of the system. The topic of this paper is the development of a methodology for selecting that decomposition of the functional design into modules that results in the least relative support cost requirements over the life cycle of the equipment.

Problem Statement

The problem addressed in this paper is that of developing a procedure which yields an optimum single level partition of a functional schematic into modules. The optimum is to be determined in the presence of several kinds of constraints. A principal determinant of logistics costs is the number of spare modules maintained at each operating site. This factor will be incorporated as a constraint on the probability that spare modules will be available when needed. This is called a spares adequacy constraint.

One of the principal purposes of modularization is to provide a rapid repair capability for equipment. This quality is commonly measured by the quantity mean-time-to-repair (MTTR). A constraint on the maximum allowable MTTR is included in the analysis.

There are often physical constraints on modules such as maximum power dissipation, weight, area and external pin count. Constraints of this type are also included in the analysis.

Although there are several types of equipments to which the procedures may be applied, emphasis is placed on electrical equipments employing microcircuit fabrication techniques.

This paper assumes a Discard at Failure maintenance philosophy. That is, the maintenance activity consists of locating failed modules, discarding the failed modules, and replacing them with new like items obtained from the spares inventory. This assumption is justified particularly for cases in which modules are constructed using microelectronic techniques.

Work is currently under way to modify the techniques presented here to accommodate multilevel partitioning problems in which both discard and repair maintenance philosophies are allowed.

One serious limitation on the procedure of this paper is the assumption that all modules of an equipment are unique. No advantage is taken of the possibility of multipurpose modules used for more than one function in an equipment. This assumption is perhaps acceptable for some types of analog equipment. It is less acceptable for digital equipment in which multipurpose modules are common. Although this limitation may be important the authors are investigating possible approaches to remove it.

Example Problem

Throughout this paper the example of Figure 1 and Table I will be used for illustration. Figure 1 represents in graphical form the functional schematic of a hypothetical equipment which is to be partitioned into modules. Each node of the graph represents an individual component or collection of components of the equipment. The collection of components represented by a node is called a smallest functional element (SFE). The SFE's are to be determined by the designer prior to initiating the analysis proposed in this paper. An SFE is a group of components which in the designer's opinion should all be packaged in the same module. For digital equipments SFE's might be gates, multivibrators, or perhaps complex combinations of components such as shift registers. During the modularization procedures SFE's will be combined to form modules, but no single SFE will be divided. SFE's serve as the basic building blocks in the synthesis of an optimum modular design.

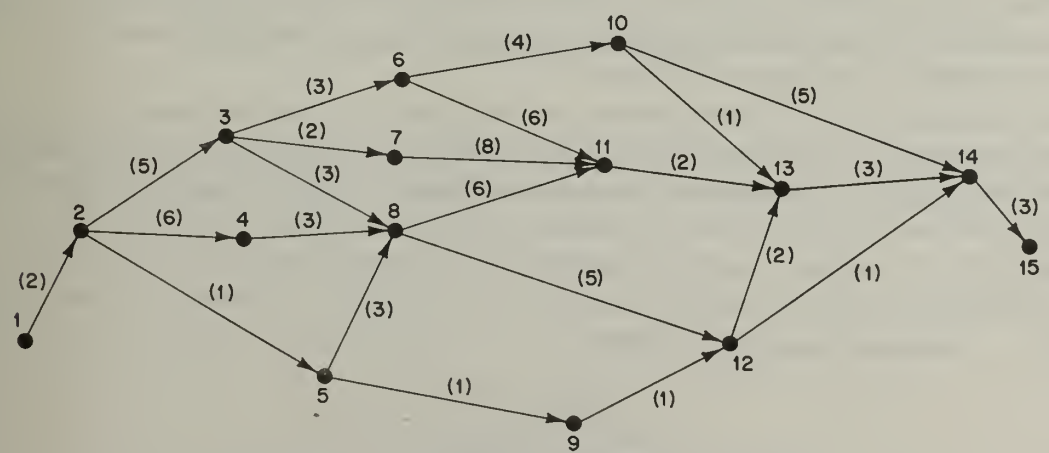


FIGURE 1. A typical functional schematic of an equipment

Nodes 1 and 15 of Figure 1 are dummy SFE's which represent the source of inputs and the destination of outputs of the equipment respectively. No modules will include these two nodes.

The lines in the figure between SFE's show the physical connections between them (these lines are subsequently referred to as arcs). The number in parentheses above an arc is the number of distinct electrical signals that must pass between the SFE's which are incident to the arc. For purposes of the analysis which follows, the arcs are given direction in such a manner that the resulting directed graph has no cycles. These directions do not necessarily represent the direction of signal flow. The assignment of directions to arcs is not in general unique and may affect the results of the analysis. Reference [5] discusses the question of this directed graph more completely. For most electrical equipment where signals flow generally from input to output the arcs should be directed in the direction of signal flow wherever possible.

Table I shows the physical characteristics of the SFE's in the example functional schematic. The maximum allowable part count, heat generation, area, and number of external pin requirements are imposed as physical constraints on the modules of the example problem.

TABLE I. *Characteristics of the SFE's of Figure 1*

SFE (i)	Part count (ϕ_i^1)	Heat generation (ϕ_i^2)	Required chip area (ϕ_i^3)	Pin count (P_i)	Failure rate/hr ($\lambda_i \times 10^{-5}$)
2	27	180	0.024	16	0.211
3	43	210	0.026	15	0.450
4	32	205	0.025	11	0.225
5	40	290	0.038	7	0.300
6	25	220	0.020	15	0.190
7	15	140	0.022	12	0.178
8	27	185	0.024	22	0.420
9	18	150	0.020	4	0.080
10	23	200	0.022	12	0.160
11	50	260	0.050	24	0.350
12	26	150	0.023	11	0.182
13	17	145	0.020	10	0.140
14	12	80	0.015	14	0.170

Representation of a Modular Design

The bold lines in Figure 2 describe a partition of the equipment into modules. Such a partition is called a design. The set of SFE's within each outlined area is to be packaged as a single module. There are three modules in the example of Figure 2.

A characteristic of the modular design problem is that there normally exists a large number of unique partitions of the functional schematic into modules. Each such partition may be identified by a positive integer index l. For example, the partition of Figure 2 may be arbitrarily defined as $l=1$. As a further definition let $n(l)$ represent the number of modules contained in partition l. It is seen that in Figure 2 $n(1)=3$. A different partition (designated $l=2$) is shown in Figure 3 for which $n(2)=4$.

The groups of SFE's within each module of the equipment will be identified by the set notation, $S_{l1}, S_{l2}, \ldots, S_{ln(l)}$. For example in Figure 2:

$$S_{1_1}=\{2, 3, 4, 5, 6, 7, 8\}$$
$$S_{1_2}=\{9, 10, 11\}$$
$$S_{1_3}=\{12, 13, 14\}.$$

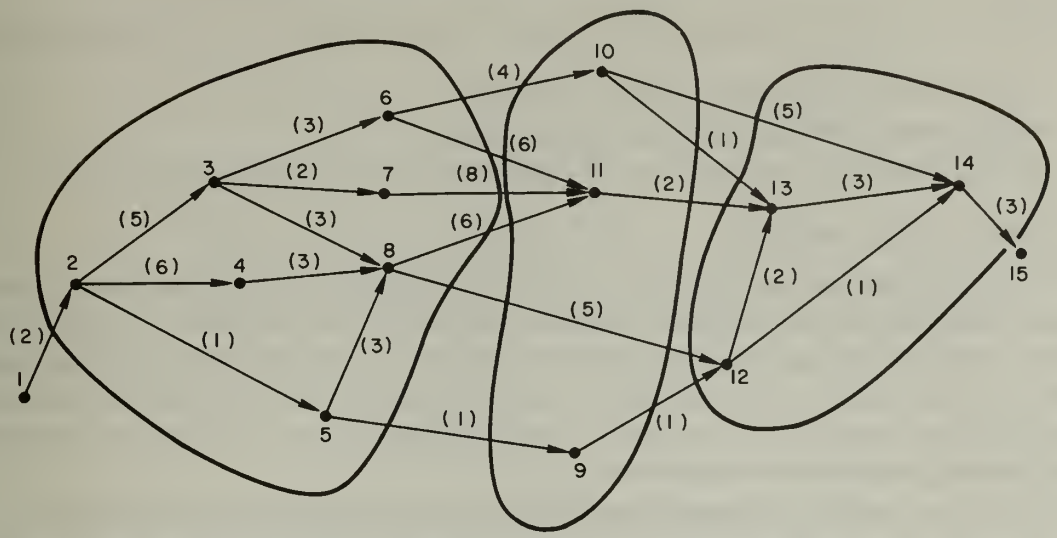


FIGURE 2. A partition of the functional schematic of Figure 1 ($l=1$)

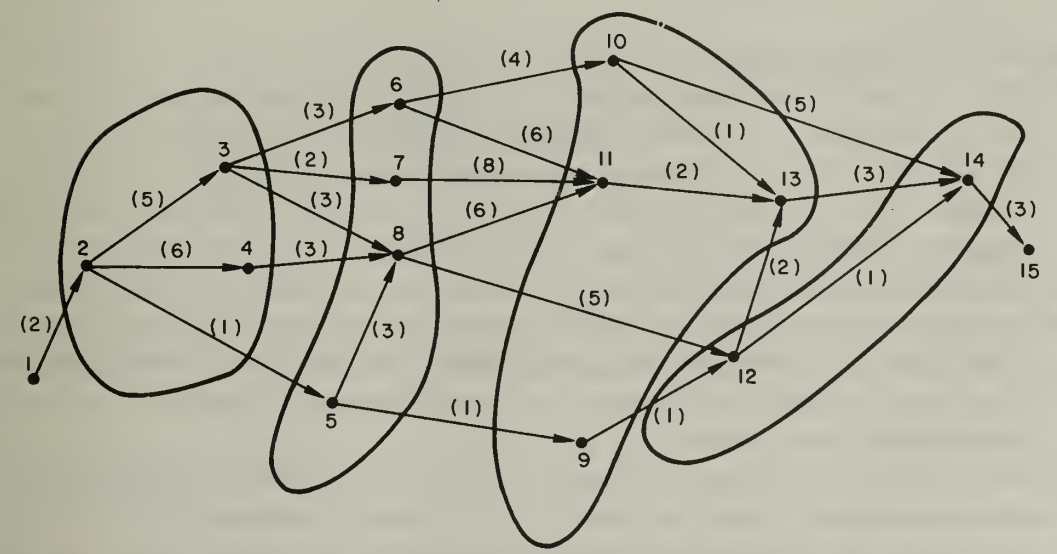


FIGURE 3. A partition of the functional schematic of Figure 1 ($l=2$)

Now suppose that it is possible to define a "life cycle cost function" $C(S_{li})$ on the subsets S_{li}. Then, again for the example of Figure 2, the life cycle cost (LCC) for the design is given by:

$$LCC = C(S_{1_1}) + C(S_{1_2}) + C(S_{1_3}).$$

The goal of this research is to find the partition l of the functional schematic such that:

$$LCC = \underset{l}{\text{minimum}} \left[\sum_{i=1}^{n(l)} C(S_{li}) \right]. \tag{1}$$

Subject to:

(2) *Inventory Constraints* (on the resulting modules S_{li})
(3) *Maintenance Constraints* (on the modular design l)
(4) *Physical Design Constraints* (on the individual modules S_{li} and the overall design l).

II MATHEMATICAL MODEL

This section contains a description of the mathematical model for the modularization problems of this paper. The model is not purported to be rigorous in the sense that it contains all of the significant elements which are necessary for an exhaustive treatment of the general microcircuit modularization problem. It is intended as a vehicle by which to demonstrate the technique which has been developed. Nevertheless, the model does contain many of the significant factors which must be considered in making sound microcircuit modularization decisions.

Certain elements of the model are reserved for the appendices because the procedures of this paper do not depend on their specific form. In particular the models which estimate the cost and reliability of modules and the model which estimates relevant maintenance time parameters are described in appendices A, B, and C, respectively.

Conditions and Assumptions

(1) The equipment is of an electronic type and a functional schematic which identifies the required components and component interconnections has been developed.
(2) The fabrication of the components into modules is to be accomplished using microelectronic construction and packaging techniques [15, 18, 20]. A classical example of such a technique is that of the TO–5 can. The TO–5 can is an individual module which (usually) contains a single, hermetically sealed integrated circuit.
(3) A discard-at-failure-maintenance (DAFM) policy is to be employed on all modules with discard occurring at the organizational level of maintenance. There are generally several possible levels of maintenance. Here only the organizational level is considered. Thus repairs are made at the operating site of the equipment.
(4) The spares provisioning is to be based on the concept of "logistic self-sufficiency" wherein adequate numbers of spares are procured at the time of acquisition to cover the expected demand over the planned equipment life cycle subject to a specified level of shortage risk.
(5) The equipment is considered to contain only a single level of indenture consisting of individual modules.

The Model Cost Equation

The model cost equation (objective function) contains the relevant cost factors subject to the conditions noted above. In a typical modularization problem (6) the cost equation is defined as follows:

$$C_T = C_A + C_S, \tag{5}$$

where:

C_T = Total Cost,
C_A = Acquisition Cost,
C_S = Lifetime Support Cost.

A further breakdown of the lifetime support costs given by (6) is represented by:

$$C_S = C_o + C_f + C_d + C_{sm}, \tag{6}$$

where:

C_o = Support cost at organizational level,
C_f = Support cost at field maintenance level,
C_d = Support cost at depot maintenance level,
C_{sm} = Support cost at special maintenance levels (e.g., factory maintenance).

Assumption (3) above eliminates all terms in Equation (6) other than C_o.
The equation simplifies to:

$$C_S = C_o. \tag{7}$$

A further definition of the support cost at the organizational level is given by:

$$C_o = C_{om} + C_{of} + C_{os} + C_{ot}, \tag{8}$$

where

C_{om} = Cost of maintenance personnel at organization,
C_{of} = Cost of consumables at organization,
C_{os} = Cost of spares at organization,
C_{ot} = Cost of transportation at organization.

The cost C_{om} is determined by maintenance manpower requirements at the organizational level. It has been pointed out by Goldman and Slattery [10] that modularized designs employing DAFM modules significantly reduce the costs attributed to maintenance. This fact is also apparent from the work reported by Andrea [1]. Goldman and Slattery also state that under a DAFM support concept, the major source of maintenance manpower costs are those associated with the time required to locate a malfunction in the equipment and the time required to remove and replace the failed module.

In a modular design the times noted above are of short duration and the direct costs associated with these times are considered to be negligible relative to the support costs of other factors in the objective function. The fact is, however, that the manpower loading exists and is incurring costs whether or not actual maintenance is being performed at any given moment. The level of the manpower which is established is classically based on equipment availability considerations [2]. A parameter which is critical to the determination of equipment availability is the equipment Mean-Time-To-Repair (MTTR).

In the current model it is assumed that the maintenance cost is based on a manpower loading level and that this level is primarily determined by a maximum allowable MTTR for the design. That is, an equipment design is permitted to achieve a specific maximum MTTR.

The cost of consumables (C_{of}) at the organizational level is considered to be negligible [6]. In any case, this factor which includes (1) the cost of utilities (power), (2) the cost of materials for maintenance of facilities, and (3) the cost of materials for maintenance of test equipment is relatively insensitive to the modular design configuration and it is therefore neglected in Equation (8).

The spares cost (C_{os}) is a significant cost factor in the modularization model. The cost of spare modules is based on the concept of "logistics-self-support" identified in condition (4). The spare

modules are considered to formally enter the inventory system at the time of equipment acquisition. The cost of the spares inventory is represented by:

$$C_{os} = n(l) \cdot C_{LI} + n(l) \cdot C_{SC} \cdot N_L + \sum_{i=1}^{n(l)} n_i \cdot C_{si}, \tag{9}$$

where

C_{LI} = Cost of introducing a line item (module type) into the supply system,
C_{SC} = Shelf cost per year of maintaining a line item in the supply system,
$\cdot n_i$ = Number of modules of type i which are procured,
C_{si} = Unit cost of module type i, and
N_L = Planned operational life of the equipment in years.

The quantities C_{LI} and C_{SC} are considered to be model constants and are regarded as input data. The parameter N_L depends on the particular equipment under consideration. C_{si} and n_i are quantities which are calculated during the course of the analysis. The module cost model which determines C_{si} is described in appendix A.

The transportation cost factor C_{ot} is reported in [6] to be negligible and is therefore not considered in the present analysis.

The acquisition cost C_A is normally considered to be made up of the following cost factors:

(1) Cost of research, design, and development,
(2) Cost of fabrication,
(3) Cost of installation,
(4) Cost of manuals,
(5) Cost of test equipment,
(6) Cost of tools and fixtures, and
(7) Cost of facilities.

Assumption (1) above establishes the existence of a functional schematic at the time of application of the current technique. At this point in the acquisition process the bulk of the research and development costs have already been incurred and are not considered as relevant cost factors at the time of modular partitioning. Any further design costs and the costs of installation, manuals, and test equipment are considered to be relatively independent of the modular design employed. The same philosophy is applied to the costs of tools, fixtures, and facilities. Fabrication cost is considered to be a relevant acquisition cost factor in the current model. This cost is expressed by the following relationship:

$$C_A = \sum_{i=1}^{n(l)} N_E \cdot C_{si}, \tag{10}$$

where

N_E = The number of equipments to be procured.

The complete objective function then becomes:

$$C_T = \sum_{i=1}^{n(l)} N_E \cdot C_s(S_{li}) + n(l) \cdot C_{LI} + n(l) \cdot C_{SC} \cdot N_L + \sum_{i=1}^{n(l)} n_i \cdot C_s(S_{li}).$$

The values of the input parameters N_E, N_L, C_{LI}, and C_{SC} used in this research were respectively 30, 10 years, \$103, and \$12. The values selected for the last two parameters are based on the work reported in [19].

Problem Constraints

The constraints imposed on the model may be classified into four distinct types: (1) the spares adequacy constraint, (2) the maintenance (MTTR) constraint, (3) partitioning constraints, and (4) physical constraints.

The Spares Adequacy Constraint

The spares adequacy constraint determines the number of module spares which must be placed in the inventory in order to achieve a specified level of adequacy. Adequacy is defined as the probability that a particular mix of module spares procured simultaneously with equipment acquisition will be adequate over the planned life of the equipment. For a given module j, $P_j(n_j)$ is defined as the probability that n_j spares of module type j will be sufficient over the planned life, N_L, of the equipment.

Assuming independence of failure among modules and a constant failure rate for individual modules, then it can be shown [3] that $P_j(n_j)$ is given by the Poisson distribution function as follows:

$$P_j(n_j) = P(w_j \leq n_j) = \sum_{w_j=0}^{n_j} \frac{e^{-\theta_j} \theta_j^{w_j}}{w_j!} \tag{11}$$

where

w_j = Total number of failures of the jth module during period N_L,
θ_j = Expected number of failures of module type j during period N_L.

θ_j may be computed from the relation:

$$\theta_j = \lambda_j \cdot N_L \cdot N_E \tag{12}$$

where

λ_j = Failure rate for the jth module type.

A model must be provided to estimate λ_j from SFE parameters. The model used in this research is described in appendix B.

For the spares package for the equipment to be adequate the spares for all modules must be adequate. Thus the adequacy of equipment spares is the product of $P_j(n_j)$ over all modules. The adequacy constraint requires that this probability be greater than some minimum acceptable value A, or

$$\sum_{j=1}^{n(l)} P_j(n_j) \geq A.$$

This inequality may be written in a more convenient form. By taking the natural logarithm of both sides and substituting the relation (11) for $P_j(n_j)$ one obtains the form of the adequacy constraint used for this analysis.

$$\sum_{j=1}^{n(l)} \ln \sum_{k=0}^{n_j} \frac{e^{-\theta j}\, \theta_j^k}{k!} \geq \ln (A). \tag{13}$$

The MTTR Constraint

Maintenance requirements are introduced into the model through a constraint on the mean-time-to-repair (MTTR). The time required to perform maintenance on a modularized equipment may be divided into four separate task times:

$$T_{\text{mnt}} = T_1 + T_2 + T_3 + T_4, \tag{14}$$

where

T_{mnt} = Time required to perform maintenance,
T_1 = Time to prepare equipment for maintenance,
T_2 = Time to isolate the failed module,
T_3 = Time to remove and replace the failed module, and
T_4 = Time to checkout the equipment after maintenance has been performed.

In general, the times required to perform each of the tasks noted above are independent random variables which obey certain underlying distribution functions. Under these conditions MTTR is given by

$$\text{MTTR} = E(T_{\text{mnt}}) = E(T_1) + E(T_2) + E(T_3) + E(T_4), \tag{15}$$

where $E(T_{\text{mnt}})$ indicates the expected value of T_{mnt}. The MTTR constraint becomes

$$E(T_{\text{mnt}}) \leq \text{MTTR}_{\text{Max}}, \tag{16}$$

where

MTTR_{Max} = Maximum allowable MTTR for the design.

The method which is used to determine the expected values $E(T_i)$ of each of the various task times is explained in appendix C. This appendix contains the development of the maintenance model for the modular design problem which is used in the current studies. From the analysis contained in appendix C the following formulation results for determining $E(T_{\text{mnt}})$:

$$E(T_{\text{mnt}}) = 2.5 + \sum_{i=1}^{n(l)} .5(.1 + .174\, e^{(.047)P(S_{li})}). \tag{17}$$

Where $P(S_{li})$ is the external pin count for module S_{li}. Given the knowledge of the pin count $P(S_{li})$ required for each of the individual modules S_{li}, $i = 1, 2, \ldots, n(l)$ contained in a modular design l, one may use Equation (17) to compute the expected value $E(T_{\text{mnt}})$.

It is acknowledged that certain of the values which were required in the development of Equation (17) were selected arbitrarily in the current analysis. This is consistent with the main objective of demonstrating how such a constraint is to be dealt with in the proposed solution methodology. In any event, it is believed that the fundamental structure of the maintenance model developed in appendix C is adequate to serve as the basis for a great many modular design studies. There is certainly nothing restricting the individual analyst from replacing the current maintenance model with one of his own choosing. However, one important restriction on the maintenance model is that it must be separable in terms of the individual modules of the design.

Physical Constraints

Certain physical constraints may be imposed on the grouping of SFE's into subsets to form modular packages. If, for example, each modular package is to contain a single microelectronic chip, there exists an upper limit on the amount of surface area that will be available. This factor must be considered in the partitioning algorithm in order to insure that only feasible modules are generated. There exist other physical constraints which may be imposed on the modules. An abbreviated list of such constraints might include

(1) Area,
(2) Total heat dissipation,
(3) Part count,
(4) Module pin count.

Constraint factors (1), (2), and (3) in the above list can be regarded as *regular constraints.* Such constraints may be incorporated in the model by defining a variable ϕ_i^k as the amount ϕ of the physical factor k which enters module j through the inclusion of SFE i. The regular constraints may then be represented by

$$\sum_{\forall i \epsilon S_{l_j}} \phi_i^k \leq \phi_{\max}^k \quad \begin{matrix}(k=1, 2, 3, \ldots, t) \\ (j=1, 2, \ldots, n(l)),\end{matrix} \tag{18}$$

where

$\phi_{\max}^k$ = Maximum allowable level of physical factor k in any individual module.
t = Total number of regular physical factors which are constrained.

The above formulation implicitly assumes that the regular constraint factors ϕ_i^k are additive.

The constraint factors for nonregular constraints cannot be modeled by a simple additive relation. An example based on pin count considerations is shown in Fig. 4. When the circuitry of modules A and B of Fig. 4 is integrated into a single module C, the interconnections a, b, c, and d between the two modules become intraconnections in module C through the integration process. The resulting module pin count decreases as a result of the integration. To account for such nonregular constraints, the following formulation is defined.

$$G^k(S_{lj}) \leq \phi_{\max}^k \quad \begin{matrix}(k=t+1, t+2, \ldots, q) \\ (j=1, 2. 3, \ldots, n(l)),\end{matrix} \tag{19}$$

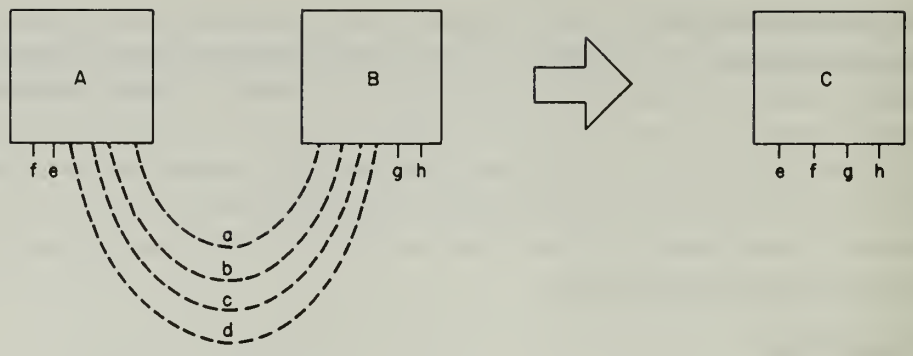


FIGURE 4. Pin count as a nonregular constraint

where

$G^k(S_{ij})$ = A function which generates the level of physical factor k established by the set of SFE's S_{ij} contained in module j.

q = The maximum number of physical factors which are constrained.

III SOLUTION PROCEDURE

Cut Partitions

The mathematical model described in the last section is a difficult one to deal with using readily available optimization techniques such as linear programming or nonlinear programming. It is possible to set up and solve the problem with integer programming, but the setup is cumbersome and the computational effort to determine a solution is great.

The procedure used is a special dynamic programming algorithm developed by Jensen to solve the general network partitioning problem. This algorithm has been modified by Caponecchi [5] to incorporate regular and nonregular constraints. The algorithm is rather complex and is described in detail in References [5, 11, 12, 13]. For this reason only a brief sketch of the procedure is presented here.

The algorithm proceeds much as a human designer might when given a functional schematic such as Figure 1. To partition this schematic into modules the designer might draw a sequence of lines across the arcs of the schematic and thus divide the schematic into distinct sets of SFE's which define the modules of the equipment. Such a set of lines and the resulting modular design is shown in Figure 5. Note that the number of pins required for a module depends on the number of signal lines which cross the designer's dividing lines. Pins have also been added to provide for bias and ground connections.

If the designer has the models described earlier and those contained in the appendices and sufficient computational power, he could probably employ trial and error to determine a good solution which meets the imposed constraints. Of course there are a large number of combinations of lines that one could draw across the schematic and the trial and error process could be difficult even with significant computational power. This is especially true if the imposed constraints are difficult to satisfy.

The procedures described here use the computational power of the digital computer and a systematic way of evaluating all possible combinations of lines of a specific type that could be drawn across the schematic. The specific lines are referred to as proper cuts and are defined below. It will be seen that the restriction to proper cuts provides the structure needed for a dynamic programming sequence,

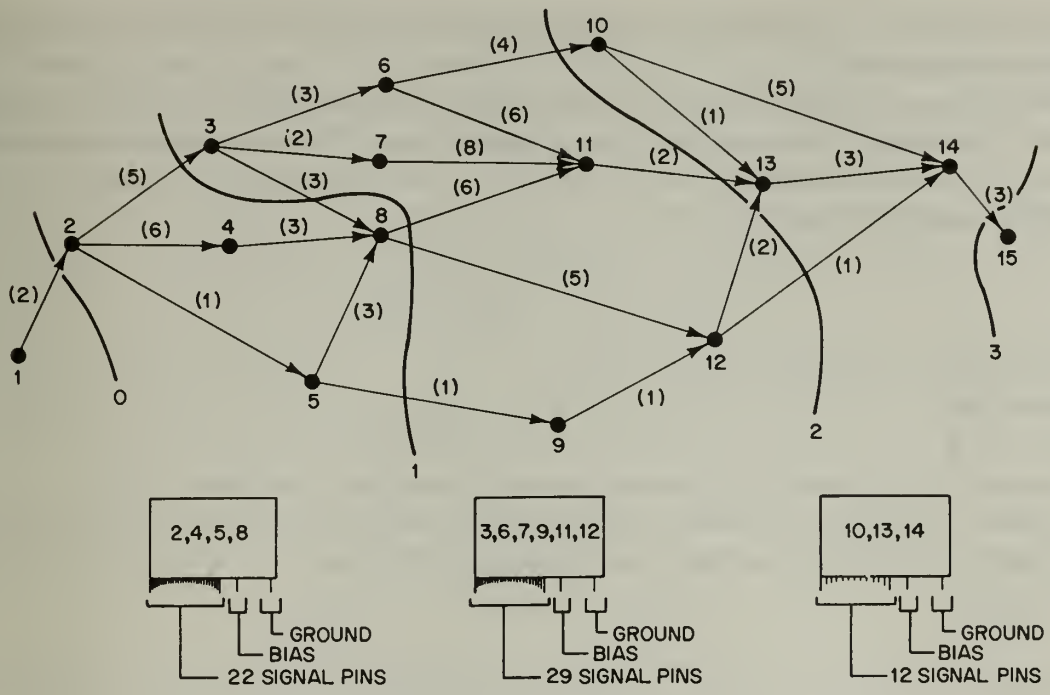


FIGURE 5. A decomposition of Figure 1 into a candidate modular design

as well as reducing the total number of module sets to a manageable sized class which can be efficiently evaluated.

In the parlance of the algorithm, the set of arcs which the designer's dividing line crosses is called a proper cut. Formally a proper cut is a set of arcs which when removed from the graph will divide it into exactly two connected subgraphs with the input node (node 1 in the example) in one subgraph and the output node (node 15) in the other. Each subgraph is connected in the sense that a path can be traced over the arcs of the subgraph (allowing both forward and reverse traversals) between every pair of nodes in the subgraph. Every proper cut i defines two sets of SFE's, the set N^i contains SFE's in the subgraph containing the input node and the set $\bar{N}^i$ contains the SFE's in the subgraph containing the output node. The proper cuts shown in Figure 5 with the corresponding sets are:

Cut i	N^i	$\bar{N}^i$
0	{1}	{2, 3, . . ., 15}
1	{1, 2, 4, 5, 8}	{3, 6, 7, 9, 10, . . ., 15}
2	{1, 2, . . ., 8, 11, 12}	{10, 13, 14, 15}
3	{1, 2, . . ., 14}	{15}.

Since nodes 1 and 15 do not represent part of the equipment, cuts 0 and 3 will always be present in such a partitioning, but there are a large variety of possible proper cuts that could be used to partition the rest of the nodes. If no other cuts were included, the design would consist of a single module including all 13 SFE's. If 12 additional cuts were used the design would consist of 13 modules each within a single SFE. For the example problem there are 187 possible proper cuts and 1,533 different modules can be specified by them. These numbers increase rapidly as the size and complexity of the

schematic increase, thus attesting to the difficulty of the trial and error exhaustive enumeration approach to this problem.

The assignment of index values of 0 thru 3 for the proper cuts shown in Figure 5 was clearly arbitrary. In general, each unique proper cut is identified by the next higher positive integer index i as each cut is generated algorithmically. The unique set of all proper cuts of a functional graph may be identified as the set C, that is

$$C = \{0, 1, 2, \ldots, i, \ldots, \rho\},$$

where the following ordering rule must be fulfilled.

If $N^i \subset N^j$,

Then $i < j$;

However If $N^i \not\subset N^j$ and $N^j \not\subset N^i$, then no relationship is implied by the indices i and j.

(The symbol $\subset$ indicates set inclusion and the symbol $\not\subset$ indicates set exclusion). Now for any particular partition l of the functional graph resulting from say $k+1$ proper cuts we may define the subset $C_l \subset C$ as follows:

$$C_l = \{l_0, l_1, l_2, \ldots, l_k\},$$

where

(1) l_i is the index of a proper cut;

(2) N^{li} denotes the set of SFE's in proper cut l_i,

(3) The indices are numbered so that $l_i < l_j \Rightarrow N^{li} \subset N^{lj}$,

(4) $\bigcup_{i=1}^{k} (N^{l}i - N^{l}i-1) = N_R$, where N_R is the complete set of SFE's of the functional schematic excluding the dummy input and output nodes.

It is seen that the index l corresponds to a particular partition of the functional schematic defined by a particular set of proper cuts.

Algorithms have been developed in this research to find the optimum partition that can be described as a collection of proper cuts [5, 11, 12, 13]. Note that the optimum thus determined is not necessarily the optimum partition of the schematic into modules, for not all partitions can be described by a set of proper cuts. A very large number of possible designs are considered however, and the idea of specifying a design as a sequence of cuts is heuristically appealing. The optimum proper cut partition does not depend on the directions assigned to the arcs in the schematic.

As the size of the schematic grows, the number of proper cuts grows rapidly, and the task of determining the optimum cut partition becomes time consuming even with a powerful computer. This has lead to an additional restriction on the class of partitions to be considered in the optimization.

A restricted cut is a cut x whose set N^x satisfies the property: if a node j is a member of N^x and an arc is directed from some node i to j in the schematic, then i must also be in the set N^x. Note that cut 1 in Figure 5 is not a restricted cut while cut 2 is. Thus the partition of Figure 5 is not a restricted cut partition. The optimum restricted cut partition is the minimum cost partition which can be described by a set of restricted cuts.

It is difficult to justify this restriction in all cases. Certainly one can propose examples in which the optimum proper cut partition is superior to the optimum restricted cut partition. However, in the

several example problems solved with both procedures during this research, the solutions determined by both methods were the same. As one heuristic justification for this restriction note that for cut 1 in the example a set of signals from nodes 2 to 3 must leave the first module and then a set of signals from 3 to 8 must reenter the module. The pins required by both sets of signals can be saved by including node 3 in the first module. The resulting cut is a restricted cut. This reduction in pin count can result in improved module reliability and a reduction in cost. In addition, the computational advantages of the restricted cut approximation are significant. This fact is indicated later in Figures 6 and 7. This approach will probably be necessary for the solution of large problems.

It should be noted that the optimum restricted cut partition may depend on the directions assigned to the arcs of the schematic graph.

Note that the subsets defining the partition can be determined from the set of cuts by the following operations. Call the partition defined by the cuts of Figure 5, then

$$S_{l1} = N^1 - N^0 = \{2, 4, 5, 8\},$$
$$S_{l2} = N^2 - N^1 = \{3, 6, 7, 9, 11, 12\},$$
$$S_{l3} = N^3 - N^2 = \{10, 13, 14\}.$$

In general, if a set of $k+1$ cuts define the sets $N^{l0}, N^{l1}, \ldots, N^{lk}$, where N^{l0} is the input node and $\bar{N}^{lk}$ is the output node, and if in addition the sets are related as

$$N^{l0} \subset N^{l1} \subset N^{l2} \ldots . \subset N^{lk},$$

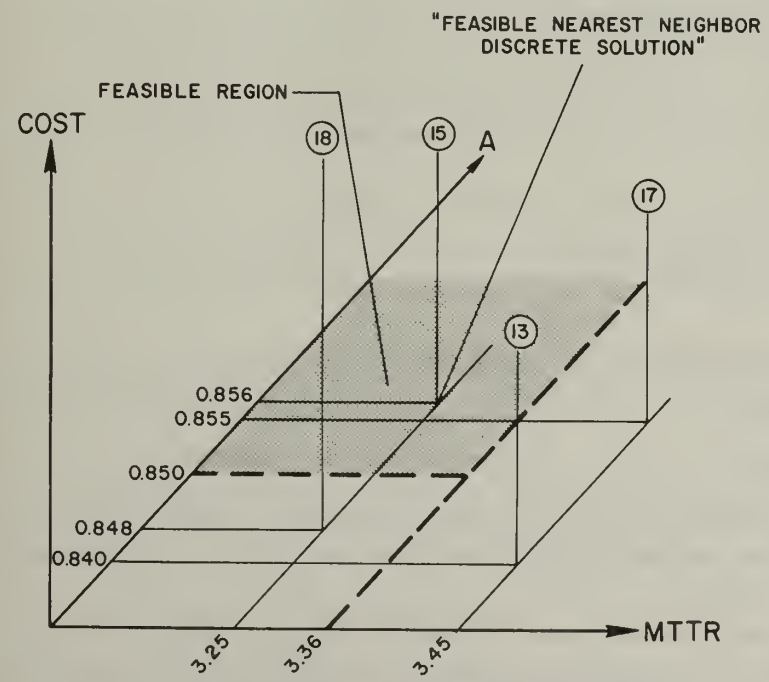


FIGURE 6. Significant results for the example problem

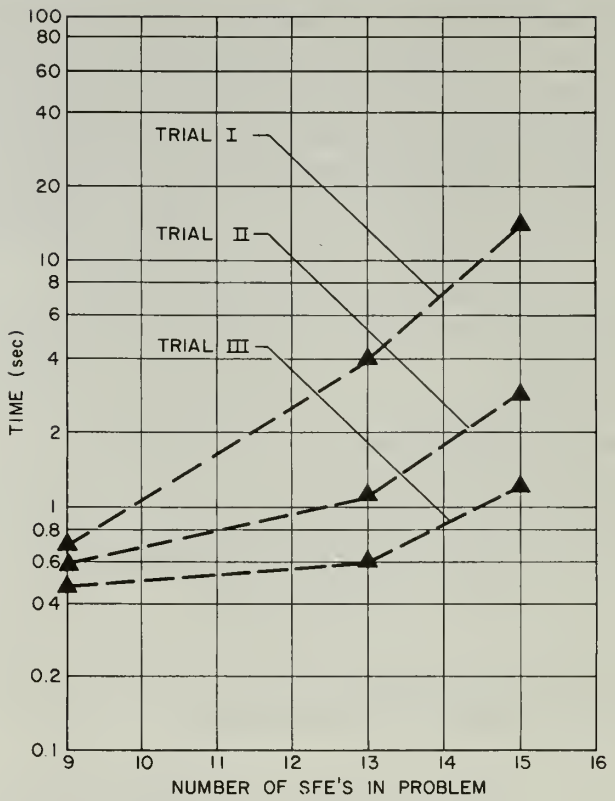


FIGURE 7. Effect of physical constraints on time required for solution (restricted cut partition)

then the following partition is defined

$$Sl_1 = N^{l1} - N^{l0}$$
$$Sl_2 = N^{l2} - N^{l1}$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot$$
$$Sl_k = N^{lk} - N^{lk-1}.$$

The set inclusion restriction is maintained because allowing overlapping cuts adds nothing to the set of possible solutions.

IV OPTIMIZATION

A dynamic programming algorithm is used to find both the optimum proper cut partition and the optimum restricted cut partition. The discussion below is applicable to both kinds of cut so the simpler term, cut, is used throughout.

Let $U(S_{li})$ be a cost function defined for all possible subsets of SFE's, S_{li}. The specific form of $U(S_{li})$ will be developed later. The models of this paper are such that the cost of the entire modularized equipment is the sum of the costs of the modules, thus

$$U_l = U(S_{l1}) + U(S_{l2}) + \ldots U(S_{l_{n(l)}}),$$

where U_l is the cost of the complete modular design l. Indeed the additivity restriction is a requirement of the dynamic programming algorithm to be described.

Since $S_{li} = N^{li} - N^{li-1}$ for two cuts defining the design, the cost of a particular module can be written

$$U(S_{li}) = U(N^{li} - N^{li-1}).$$

Consider that portion of the schematic defined by the SFE's in the set N^j for some cut j. It is possible to determine the optimum partition of the set N^j independently of the SFE's in $\bar{N}^j$ because of the additivity of the cost function. Call the cost of the optimum partition of the set N^j, V_j.

Consider now a cut i defining the set of SFE's, N^i. How can one determine the optimum partition of this set of SFE's? Let $N^j \subset N^i$, thus a module can be defined by the difference $N^i - N^j$. The cost

$$U(N^i - N^j) + V_j,$$

is the cost of the partition of the set N^i into the module $N^i - N^j$ and the modules in the optimum partition of N^j.

The optimum partition of N^i can be found by calculating this cost for all cuts which define sets which are included in N^i and choosing the combination yielding the minimum cost. Define P_i to be the set of all cuts such that if $j \epsilon P_i$ then $N^j \subset N^i$. Then the cost of the optimum cut partition for N^i is

$$V_i = \min_{j \epsilon P_i} [U(N^i - N^j) + V_j]. \tag{20}$$

This is a typical dynamic programming recursive equation. It can be solved by noting that $V_o = 0$ where N^o is the subset consisting of only the input node. If the cuts are indexed so that $N^j \subset N^i$ implies that $j < i$, the equation can then be solved for $i = 1$, and then $i = 2$ and so on. The optimum solution is found when the equation is solved for cut ρ, where ρ is the cut with the greatest index and $\bar{N}^\rho$ consists only of the output node.

The Jensen algorithm performs all the required functions to affect the complete optimization for any given schematic, including the generation of all cuts, the indexing of cuts in proper order, the solution of the recursive equation above for all cuts, and finally the back-tracking procedure required to recover the optimum design.

The user is only required to define the cost function $U(S_{li})$ defined for all possible modules.

Solution of the Modularization Problem

The objective function and constraints of the modularization problem have been stated and an algorithmic procedure has been described which yields the optimum proper or restricted cut partition. It now remains to combine these two aspects of this research to obtain solutions to the modularization problem.

The mathematical model of the modular design problem as developed earlier is

$$\min_l C_T = \sum_{i=1}^{n(l)} \{C_{LI} + C_{SC} \cdot N_L + N_E \cdot C_s(S_{li}) + n(S_{li}) \cdot C_s(S_{li})\}, \tag{21}$$

subject to

$$(22) \qquad \text{(Adequacy Constraint)} - \sum_{i=1}^{n(l)} \ln \sum_{k=0}^{n(S_{li})} \frac{e^{-\theta(S_{li})}\theta^k(S_{li})}{k!} \leq -\ln(A) \quad (\theta(S_{li}) = \lambda(S_{li}) \cdot N_E \cdot N_L),$$

$$(23) \qquad \text{(MTTR Constraint)} \; E(T_{\text{mnt}}) \leq \text{MTTR}_{\max},$$

$$(24) \qquad \text{(Regular Physical Constraints)} \sum_{\forall i \in S_{lj}} \theta_i^k \leq \theta_{\max}^k \qquad \begin{matrix}(k=1, 2, 3) \\ (j=1, 2, \ldots, n(l)),\end{matrix}$$

$$(25) \qquad \text{(Nonregular Physical Constraint)} \; G^4(S_{li}) \leq \theta_{\max}^4 \quad (i=1, 2, 3, \ldots, n(l)).$$

The notation used in Equations (21) and (22) is identical to that introduced previously. The following comments apply to the notation used in the remaining equations.

ϕ_i^1 = Number of component parts in SFE i which is contained in module type N_{lj}.

$\phi_{\max}^1$ = Maximum allowable number of parts in any given module.

ϕ_i^2 = Amount of heat (Watts) generated by the components of SFE i which is contained in module type S_{lj}.

$\phi_{\max}^2$ = Maximum amount of heat dissipation permitted in any given module.

ϕ_i^3 = Chip surface area required by the components which make up SFE i, which is contained in module type S_{lj}.

$\phi_{\max}^3$ = Maximum allowable chip surface area in any given module.

$G^4(S_{li})$ = Number of external pin connections required by the subset of SFE's which make up module type S_{lj}.

$\phi_{\max}^4$ = Maximum allowable number of external pin connections for any given module.

The GLM Solution Technique

The mathematical model stated above is not in a form directly applicable to the Jensen algorithm. In particular the adequacy constraints (22) and the MTTR constraint (23) are system constraints rather than constraints on individual modules. When constraints of this type are imposed on the problem solution, state variables arise in a dynamic programming formulation and result in increased dimensionality of the problem.

To circumvent the problem of state variables the concept of Generalized Lagrange Multipliers (GLM) is employed. The GLM technique permits the incorporation of certain constraints into the objection function, thus eliminating the state variables which would otherwise appear if the constraints were considered separately.

Constraints (22) and (23) are handled by multiplying the left side of each constraint by a multiplier, λ_1 and λ_2, respectively, and adding the result to the objective. Thus the objective function

becomes

$$\min Z = C_T - \lambda_1 \sum_{i=1}^{n(l)} \ln \sum_{k=0}^{n(S_{li})} \frac{C^{-\theta(S_{li})}\,\theta^k\,(S_{li})}{k!} + \lambda_2\, E(T_{\text{mnt}})\,. \tag{26}$$

The corresponding constraints are then eliminated from the problem.

The solution procedure starts with an arbitrary choice of λ_1 and λ_2. The new minimization problem is solved to obtain the optimum partition. The values of adequacy and MTTR are calculated for this solution. If they are close enough to the desired values the procedure is terminated. Otherwise new λ's are chosen and the problem is solved again.

The multipliers λ_1 and λ_2 can be viewed as penalties for the constrained quantities. If the λ's are too low, the constraints will not be satisfied. If the λ's are too high, the constraints will be satisfied in excess.

The trade-off of the GLM technique is that a single pass solution is replaced with an iterative solution. However, the problem which must be solved at each iteration is simpler than the original problem in that the MTTR and adequacy constraints are no longer explicitly considered.

The problem of choosing the multipliers λ_1 and λ_2 for subsequent iteration is a complicated one. The procedures developed automatically adjust the multipliers using a method suggested by Brooks and Geoffrion [4]. The authors intend to present a subsequent paper which deals with this specialized topic.

Computational Results

To demonstrate the computational characteristics of the solution procedure a problem was formulated based on the functional graph of Figure 1 and the SFE data of TABLE I. The levels of the resources which were selected for each of the constraints are indicated in the formal statement of the problem as shown below:

$$\min_{l} C_T = \sum_{i=1}^{n(l)} \{C_{LI} + C_{SC} \cdot N_L + N_E \cdot C_s(S_{li}) + n(S_{li}) \cdot C_s(S_{li})\},$$

subject to

$$-\sum_{i=1}^{n(l)} \ln \sum_{k=0}^{n(S_{li})} \frac{e^{-\theta(S_{li})}\theta^k(S_{li})}{k!} \leqslant -\ln\,(0.850)$$

$$(\theta(S_{li}) = \lambda(S_{li}) \cdot N_E \cdot N_L)$$

$$2.5 + \sum_{i=1}^{n(l)} 0.5\;(0.1 + (0.174)e^{0.047 \cdot P(S_{li})} \leqslant 3.360$$

$$\sum_{\forall i \in S_{lj}} \phi_i^1 \leqslant 150 \qquad (j = 1, 2, \ldots, n(l))$$

$$\sum_{\forall i \in S_{lj}} \phi_i^2 \leqslant 1100 \qquad (j = 1, 2, \ldots, n(l))$$

$$\sum_{\forall i \epsilon S_{lj}} \phi_i^3 \leq 0.17 \qquad (j=1, 2, \ldots, n(l))$$

$$\sum_{\forall i \epsilon S_{lj}} P_i - \Delta P(S_{lj}) \leq 25 \qquad (j=1, 2, \ldots, n(l)),$$

where

1. The parameters $C_S(S_{li})$ and $\lambda(S_{li})$ are obtained by use of the supporting models described in appendix A and B.
2. The values assigned to the various model constants are as indicated in section II and in the appendices.
3. The functional graph is that of the example problem, Figure 1.
4. P_i is the number of external pins required by SFE_i in module S_{lj}, $\Delta P(S_{lj})$ is the number of external pins eleminated by placing the SFE's in the single module S_{lj}.

In order to apply the dynamic programming optimization procedure described previously, the problem is first transformed into the GLM form described by Equation (26):

$$(27) \qquad \min_{l} \sum_{i=1}^{n(l)} \Bigg\{ C_{LI} + C_{SC} \cdot N_L + N_E \cdot C_S(S_{li}) + \min_{n(S_{li})} \left[n(S_{li}) \cdot C_S(S_{li}) - \lambda_1^k \ln \sum_{j=0}^{n(S_{li})} \frac{e^{-\theta(S_{li})} \theta^j(S_{li})}{j!} \right] + \lambda_2^k [0.5(0.1 + (0.174) e^{0.047 \cdot P(S_{li})})] \Bigg\},$$

subject to

$$(28) \qquad \sum_{\forall i \epsilon S_{lj}} \phi_i^1 \leq 150 \qquad (j=1, 2, \ldots, n(l))$$

$$(29) \qquad \sum_{\forall i \epsilon S_{lj}} \phi_i^2 \leq 1100 \qquad (j=1, 2, \ldots, n(l))$$

$$(30) \qquad \sum_{\forall i \epsilon S_{lj}} \phi_i^3 \leq 0.17 \qquad (j=1, 2, \ldots, n(l))$$

$$(31) \qquad \sum_{\forall i \epsilon S_{lj}} P_i - \Delta P(S_{lj}) \leq 25 \qquad (j=1, 2, \ldots, n(l)).$$

The constrained optimum to this problem is determined by repetitively solving the following recursive equations of dynamic programming for different selected values of λ_1^k and λ_2^k:

$$(20) \qquad V_i = \min_{j \epsilon P_i} [U(N^i - N^j) + V_j],$$

where $j \epsilon P_i$, if $N^j \subset N^i$ and if:

$$(32) \qquad \sum_{\forall i \epsilon (N^i - N^j)} \phi_i^k \leq \phi_{\max}^k \qquad (k=1, 2, 3)$$

(33) $$\sum_{\forall i \in (N^i - N^j)} P_i - \Delta P(N^i - N^j) \leq \phi^4_{\max}$$

And the cost function at each stage is defined as follows:

(34) $$U(N^i - N^j) = C_{LI} + C_{SC} \cdot N_L + N_E \cdot C_S(N^i - N^j) + \min_{n(N^i - N^j)} \left[n(N^i - N^j) \cdot C_S(N^i - N^j) - \lambda_1^k \ln \sum_{j=0} \frac{e^{-\theta(N^i - N^j)} \theta^{j(N^i - N^j)}}{j!} \right] + \lambda_2^k [.5(.1 + (.174)e^{1047 \cdot P(N^i - N^j)}].$$

The problem described by the relations (27)–(31) is solved initially for an arbitrary choice of multipliers λ_1^1 and λ_2^1. Subsequent values of these multipliers are determined by linear approximations using the GLM–LP technique suggested by Brooks and Geoffrion [4].

A typical solution sequence based on restricted cut partitioning is shown in Table II. This solution sequence resulted for the example problem with an initial choice of multipliers: $\lambda_1^1 = 20{,}000$ and

TABLE II. *Solution to the Example Problem (Restricted Cut Decomposition)*

$\lambda_1^1 = 20{,}000$, $\lambda_2^1 = 7{,}000$

$A = 0.85$			$MTTR_{\max} = 3.3600$		
Iteration (k)	Value of lagrange multipliers		Minimum cost obtained ($C_T \times 10^{-5}$)	Level of adequacy (A^k)	Mean time to repair ($MTTR^k$)
	($\lambda_1^k \times 10^{-4}$)	($\lambda_2^k \times 10^{-4}$)			
1	2.00	0.70	2.13	0.676	3.87
2	254.82	0.00	2.62	0.999	3.87
3	0.13	25.76	2.44	0.951	2.25
4	0.00	5.03	0.34	0.000 [a]	2.25
5	0.04	5.01	1.10	0.000	2.25
6	0.12	4.96	3.25	0.000	2.25
7	0.33	4.84	2.02	0.008	2.25
8	0.90	4.53	2.10	0.336	3.45
9	2.53	3.63	2.22	0.753	3.45
10	9.46	0.00	2.25	0.939	3.87
11	6.95	2.91	2.30	0.915	3.45
12	4.07	6.26	2.36	0.833	3.25
13	4.07	4.73	2.25	0.840	3.45
14	5.18	5.24	2.38	0.876	3.25
15	4.46	5.22	2.37	0.856	3.25
16	4.78	5.15	2.27	0.870	3.45
17	4.37	5.19	2.26	0.855	3.45
18	4.18	5.21	2.36	0.848	3.25
19	4.13	5.20	2.36	0.840	3.25
20	4.14	5.20	2.36	0.848	3.25

[a] $A \leq 10^{-5}$ is reported as $A = 0.000$.

$\lambda_2^1 = 7{,}000$. The more significant results of this solution sequence are shown plotted in Figure 6. The results at iterations 13, 15, 17, and 18 constitute what are termed the "nearest neighbor discrete solutions" to the problem.

It is seen in the figure that no discrete solution exists that satisfies with equality the specified constraints on adequacy and MTTR $_{max}$. Note also that the proposed solution technique which is based on the Jensen Algorithm provides a family of solutions which bracket the desired constraint resource levels. The solution generated at iteration 15 (which corresponds to $\lambda_1^{15} = 44,588$ and $\lambda_2^{15} = 52{,}180$) represents the only "feasible nearest neighbor discrete solution" obtainable by the GLM solution procedure. The modular design which resulted for this solution is shown in Table III.

It is of interest to note that the tabulation of solutions as indicated in Table II provides the analyst with the inherent capability to carry out a direct sensitivity analysis based on the resource levels of the system constraints. For example, the data associated with the result at iteration 18 indicate the effect on optimal cost of a slight reduction in the adequacy constraint.

In the course of this research two other example problems were formulated. One of these problems contained 9 SFE's and the other problem contained 13 SFE's. Computational trials were carried out on a CDC 6600 computer with this family of three problems to provide insight into the following questions:

(1) What is the effect of the number of SFE's on the computation time required to obtain a solution?
(2) What effect do the physical constraints have on the time required to obtain a solution?
(3) Does proper cut partitioning significantly increase the time required to obtain a solution?
(4) Are the optimal results obtained by proper cut partitioning identical to those obtained by restricted cut partitioning?

The answers to questions (1), (2), and (3) for the three specific problems which were studied are revealed in Figures 7 and 8.

TABLE III. *Optimal Design for Example Problem at Iteration 15*

$(\lambda_1^{15} = 44{,}588, \lambda_2 = 52{,}180)$

Modules appearing in the optimal design (S_{ti})	Number of spares ($n(S_{ti})$)	Number of pins ($P(S_{ti})$)	Part count (θ_i^1)	Heat generation (θ_i^2)	Required chip area (θ_i^3)
(6, 7, 10, 11, 13, 14)	152	19	56	1045	0.149
(5, 8, 9, 12)	212	18	102	770	0.102
(2, 3, 4)	186	16	102	595	0.075

$$A = 0.856$$
$$\text{MTTR} = 3.25$$

In order to obtain these figures each of the problems was solved in one iteration using appropriate values of λ_1 which resulted in min. C_T for $A \geq 0.85$. The value of λ_2 was selected at $\lambda_2 = 0$. Each of the example problems were solved three times for each type of cut partition. For each time trial run a

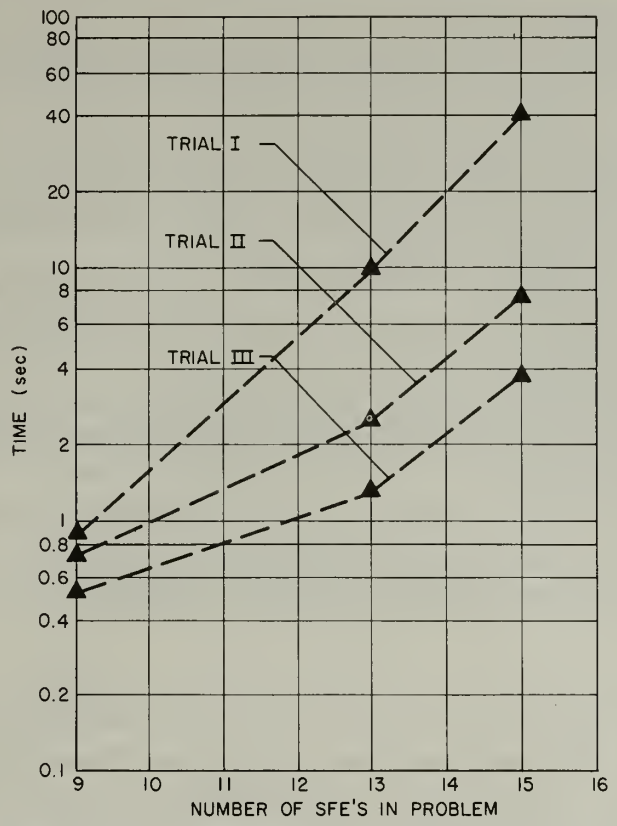


FIGURE 8. Effect of physical constraints on time required for solution (proper cut partition)

tighter set of physical constraints were imposed on the problem solution. The constraint sets which were used are indicated in Table IV.

An analysis of the constraint levels contained in the table reveals that the choice of the level in Trial I results in essentially a physically unconstrained problem for this trial.

The results obtained in the trial runs for both proper and restricted cut partition of the functional graphs are indicated in Figures 7 and 8. In these figures the time required to obtain the optimum design is plotted versus the number of nodes in the particular problem under consideration.

TABLE IV. *Physical Constraints Used in Each Time Trial Run*

Constraint	Constraint parameter	Level of physical constraints		
		Trial I	Trial II	Trial III
Part count	θ^1_{max}	10,000	150	100
Heat generation	θ^2_{max}	10,000	800	650
Required chip area	θ^3_{max}	1,000	0.16	0.10
Pin count	θ^4_{max}	10,000	35	25

The effect of the physical constraints in reducing the time required to obtain a problem solution is clearly evident in these figures. The obvious conclusion is that as one imposes more constraints on the modular design problem and/or tightens the respective bounds of the existing constraints, the required time to achieve a solution may be expected to decrease. This result is contrary to the normal course of events in most optimization problems and is indeed fortuitous in the present case. One would certainly expect to encounter a reasonable number of regular and nonregular physical constraints in an actual application of the solution procedure.

The time required to obtain a solution appears to increase in an exponential-like fashion with respect to the number of SFE's in the problem. This result corresponds with an earlier, more thorough analysis conducted by Jensen [11].

It can be seen that the use of proper cuts resulted in approximately a three-fold increase in the time required to obtain a solution to the two largest problems. This increase was not as marked for the smallest of the three problems.

A result which is not revealed by Figures 7 and 8 is that the restricted cut and proper cut optimal designs obtained were identical for each of the three example problems. This result is encouraging when one considers the superior computational efficiency provided by restricted cuts.

V CONCLUSIONS

The following conclusions can be drawn from the research conducted to date:

(1) The Jensen Algorithm provides a systematic and logical means for implicitly evaluating a significantly large number of unique candidate modular designs generated by proper or restricted cuts of the functional graph of an equipment. In addition, the algorithm employs an efficient optimization procedure for selecting the optimum design based on the cost criteria imposed on the design problem.

(2) As expected, solution by restricted cut partition of the functional graph was determined to be more efficient than solution by proper cut decomposition. Efficiency as used in this context refers to the computer time required to obtain an optimal solution and to the required computer storage capacity.

(3) Although in principle restricted cut partition of the functional graph represents a more severe limitation on the optimality of the results obtained than does proper cut partition, the *same* optimal design resulted from the use of *both* types of cut partition. This finding was observed to hold for all of the example problems which were solved.

(4) The presence of physical design contraints was found to be *beneficial* to the proposed solution procedure. Such constraints resulted in a *decrease* in the computer time required to obtain solutions to the modular design problem when utilizing either proper or restricted cuts of the functional graph.

APPENDIX A

Module Cost Model

Cost models for microelectronic circuitry are the subject of on-going research and some have been reported in the literature. Evans [7] states that:

. . . The cost of an integrated circuit depends on its area and it is therefore important to know the areas required for various electronic functions when achieved by the different integrated circuit processes.

The company's experience with bipolar circuits has shown that for these the area can be calculated from a circuit diagram. Tables and graphs exist showing the areas of individual components which simply have to be added together; this total is then increased by 20 percent to allow for cross-over and wasted area due to non-ideal layout and the final figure is very close to what can be achieved in an actual layout . . ..

Cost information of the type specified by Evans can easily be included in the technique to be developed here. Cost models are also discussed by Johnson [14] and in Reference [20].

Detailed cost models are usually proprietary and are not generally available. A linear cost model based on the comments contained in [7] has been used in the current research effort. The cost model selected in this research was as follows

$$C(S_{li}) = C_1 \cdot V(S_{li}) + C_2 \cdot P(S_{li}) + C_3,$$

where

C_1 = Component Cost Factor,

C_2 = Interconnect Cost Factor,

C_3 = Package Cost Factor, and

$V(S_{li})$ = Number of Components in Module S_{li}.

The respective values assigned to C_1, C_2, and C_3 in the current work were \$2, \$2, and \$100.

APPENDIX B

Module Reliability Model

The failure rates of the feasible modules which are formed when the modularization technique is applied to the functional schematic are a critical parameter in the optimization process. For modularization problems involving discrete components, it is generally assumed that failure rates may be added.

Reference [20] contains a general discussion of the data requirements which are necessary in order to construct a meaningful microelectronic circuit failure model. The existence of such a model is of great importance if practical microcircuit partitioning techniques are to be developed. One of the most significant studies of this nature has just recently been published by the Boeing Company [17].

Great flexibility exists as to the type of reliability model which can be incorporated into this procedure. This study estimates the module failure rate as a simple sum of the failure rates of the component SFE's of the module less a correction factor for external connections eliminated by placing more than one SFE in a package. The formula used in the current work for computing the failure rate $\lambda(S_{li})$ of the module S_{li} is

$$\lambda(S_{li}) = \sum_{\forall j \in S_{li}} \lambda_j - \Delta P(S_{li}) \cdot \lambda_p,$$

where

λ_j = Failure rate of SFE j contained in module S_i,

λ_p = Failure rate correction factor for interconnect reduction $(.1 \times 10^{-5})$.

It should be noted that the failures of SFE's on the same chip may well be dependent. For example, a scratch on a chip incurred say, during repair of an adjacent chip might cause a dependent failure. If all such dependent failure mechanisms were included in the individual SFE failure rates, the results found here would be pessimistic values.

APPENDIX C

Maintenance Model

To utilize the modular design procedure, some model must be provided to estimate the expected repair time for the system. This model is required for constraint Equation (23). This equation is reproduced below for ease of reference.

$$E(T_{\text{mnt}}) \leq \text{MTTR}_{\text{max}}, \tag{23}$$

where

$$T_{\text{mnt}} = T_1 + T_2 + T_3 + T_4. \tag{14}$$

The task times T_i are to be regarded as independent random variables which satisfy certain underlying distribution functions. From considerations of fundamental probability theory [21], Equation (23) may be rewritten as follows:

$$E(T_1) + E(T_2) + E(T_3) + (T_4) \leq \text{MTTR}_{\text{max}}. \tag{35}$$

The following assumptions are applicable to the further development of Equation (35).

1. Failure of the equipment results from the failure of a single module within the equipment.
2. Corrective maintenance consists of: (1) preparing the equipment for fault location, (2) fault locating the failed module, (3) obtaining a spare of the failed module from the inventory and replacing it in the failed equipment, and (4) performing a standard checkout procedure on the maintained equipment to insure that it is operational.
3. A spare module is always available to replace a failed module (i.e., no backorders ever occur).
4. Fault location of the modules in a failed equipment is accomplished by testing the individual modules in a random sequence.
5. The time required to checkout a specific module is exponentially related to the module complexity.
6. The distributions of the random variables T_1, T_3, and T_4 are independent of the modular configuration of the equipment.

Assumption (1) is commonly made in many maintainability/reliability analyses. Assumption (2) represents what is believed to be a reasonable structure for the maintenance model of a modular equipment. A similar model structure may be found in the studies reported by Philco/Ford [6].

A detailed analysis of T_2, the time to isolate the failed module was carried out in Reference [6], and it was determined that the time to fault locate a modular system is exponentially related to module complexity (Assumption 5.). For the purposes of the current study, module pin count has been se-

lected as an index of module complexity. The functional relationship which has been selected is as follows:

$$t_{2i} = t_1 + k_1 e^{k_2 P(S_{li})}, \tag{36}$$

where

t_{2i} = Time required to checkout module i.

t_1 = Time required to prepare a module for checkout (assumed to be constant, independent of module complexity). A value of 0.1 hr was used for this parameter.

k_1, k_2 = Furnished model constants. Respective values of 0.174 hr and 0.047 hr were used.

$P(S_{li})$ = Number of external pin connections required by module i.

The rationale behind the assumption indicated in (4) above is as follows:

a. Skill level and/or preferences of an individual maintenance technician will have a bearing on the sequence in which the modules are tested.

b. The modular design analysis is conducted early in the design phase of the equipment acquisition process, and it is therefore very unlikely that a fault symptom matrix has as yet been developed. A symptom matrix would necessarily be dependent on the final design configuration and would have a definite effect on the recommended sequence for module fault testing.

c. The results obtained by a random selection sequence are conservative in that they will result in an overestimate of the required fault location time.

In view of these considerations, it is believed that an analysis based on a random module selection sequence will yield reasonable estimates of MTTR.

If one assumes a random test sequence, then on the average approximately half the modules must be tested before the failed module is found [5]. Thus the fault isolation time can be estimated as

$$E(T_2) = 0.5 \sum_{i=1}^{n(l)} t_{2i}. \tag{37}$$

The values assigned to the expected values T_1, T_3, and T_4 in this work were 1 hr, 1 hr, and 0.5 hr, respectively. Substituting the indicated values in Equations (35), (36), and (37) yields:

$$E(T_{\text{mnt}}) = 2.5 + \sum_{i=1}^{n(l)} .5(.1 + .174e^{(.047)P(S_{li})}). \tag{17}$$

BIBLIOGRAPHY

[1] Andrea, J. J. "An Evaluation of Module Replacement and Disposal at Failure for Maintenance of Ground Electronic Equipment," RADC–TR–60–5, Rome, N.Y.: Rome Air Development Center (1960), AD 232917.

[2] Barton, H. R., *et al.*, *A Queuing Model for Determining System Manning and Related Support Requirements.* TDR No. AMRL–TDR–64–21. Wright-Patterson AFB, Ohio: Aerospace Medical Research Laboratories (Jan. 1964), AD 434803.

[3] Bazovsky, I., *Reliability Theory and Practice* (Prentice-Hall Inc., New Jersey, 1961).

[4] Brooks, R. and Geoffrion, A., "Finding Everett's Lagrange Multipliers by Linear Programming," Opns. Res. *14*, 1149–1153 (Nov.–Dec. 1966).

[5] Caponecchi, A. J., "A Methodology for Obtaining Solutions to the Modular Design Problem," unpublished Ph. D. dissertation. The University of Texas at Austin, Austin, Texas (Dec. 1971).

[6] Day, H. W. and Jordan, P., "Maintainability of Microcircuit Equipment," RADC–TR–68–187, Rome, N.Y.: Rome Air Development Center (1968), AD 842399.

[7] Evans, J. D., "Integrated MOST Circuits," *Microelectronics and Reliability 7*, 1–36 (Feb. 1968).

[8] Everett, H., "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," Opns. Res. *11*, 399–417 (May–June 1963).

[9] Goldman, A. S., "Problems in Life Cycle Support Cost Estimation," Nav. Res. Log. Quart. *16*, 111–120 (Mar. 1969).

[10] Goldman, A. S. and Slattery, T. B., *Maintainability* (John Wiley and Sons, New York, 1964).

[11] Jensen, P. A., "Optimum Network Partitioning," to appear in Opns. Res.

[12] Jensen, P. A., "An Implicit Enumeration Scheme for Proper Cut Generation." Technometrics *12*, 775–787 Nov. 1970).

[13] Jensen, P. A., "A Graph Decomposition Technique for the Design of Reliable Redundant Electronic Networks," unpublished Ph. D. dissertation, Johns Hopkins University, Baltimore, Md. (Sept. 1967).

[14] Johnson, H. "Anatomy of Integrated Circuit Technology," IEEE Spectrum *7*, 56–66 (Feb. 1970).

[15] Merrigan, M. A., *et al.*, *Handbook of Design Criteria for Microelectronic System Packages.* RADC–TR–67–125. Rome, N.Y.: Rome Air Development Center (1967), AD 655–762, –763, 764, and –765.

[16] Morris, T. D., "Upgrading the Effectiveness of Logistics Systems." Defense Management Journal, Vol. III, No. 4 (1967).

[17] Porter, D. C. and Finke, W. A., "Reliability Characterization and Prediction of Integrated Circuits," The Boeing Co., Aerospace Group. RADC–TR–70–232. Rome Air Development Center (Nov. 1970).

[18] Radio Corporation of America, "Introduction to the Design and Application of Micromodules," U.S. Army Electronics Command, Contract No. DA–36–039–SC–75968 (Sept. 1963), AD 639590.

[19] U.S. Department of the Air Force, Air Force Logistics Command. "Optimum Repair Level Analysis," AFLCM/AFSCM 375–6. Wright-Patterson AFB, Ohio. (May 1968).

[20] U.S. Naval Applied Science Laboratory. *Handbook for Systems Application of Redundancy.* Washington, D.C.: U.S. Naval Applied Science Laboratory (1966), AD 804277.

[21] Wine, L. R., *Statistics for Scientists and Engineers* (Prentice Hall, Inc., New Jersey, 1964).

SURVIVAL PROBABILITIES ASSOCIATED WITH CROSSING FIELDS CONTAINING ABSORPTION POINTS*

James A. Parsons

Lederle Laboratories
Pearl River, N.Y.

INTRODUCTION

Consider a situation where a particle must cross a path which takes it through a region which contains points, termed *absorption points*, such that if the particle contacts one of these points it will be absorbed there with probability p, $0 \leq p \leq 1$. If a particle is absorbed at an absorption point then its progress across the path is halted, and also this particular absorption point is ruined in the sense that it cannot absorb any future particles. It is assumed that absorption points act independently.

Using combinatorial arguments, Zacks and Goldfarb [1], developed an expression for the distribution of the number of particles absorbed among the first n ones that attempt to travel the path. This is the conditional *pdf* $Pr[D_n = k/R = r]$ where, for $n = 1, 2, 3, \ldots$, D_n is the number of particles absorbed out of the first n that try to cross, given R, the number of absorption points in the path when the very first particle began to cross. Using this distribution, certain survival distributions can be developed through the application of formal rules covering compound events.

In this paper a different, and computationally more convenient, expression for $Pr[D_n = k/R = r]$ is developed through the use of basic probability arguments. First a *pdf* will be established from a generalized Bernoulli scheme, which itself appears to have general applications, and then, through the specification of the parameters of this *pdf* the expression we seek will be obtained automatically.

A GENERALIZED BERNOULLI SCHEME

What we have in mind here is a generalization of the scheme of Bernoulli trials: one which allows the probability of a success on each trial to depend on the number of previous successes up to this trial.

Enroute to the final *pdf* of the probability of k successes in n consecutive trials we make use of the following:

LEMMA: If the probability generating function $G(k, t) \equiv \sum_{j=0}^{\infty} P_j(k) \cdot t^j$, $|t| < 1$ (with $P_j(k) \equiv Pr[X_j = k]$), has the following form:

$$\sum_{j=0}^{k} g_j(k) \cdot [(ta_j)^k \cdot (1 - ta_j)^{-1}], \tag{1}$$

where $g_j(k)$ is independent of t and $|a_j| < 1$, then:

$$P_n(k) = \sum_{j=0}^{k} g_j(k) \cdot (a_j)^n. \tag{2}$$

*This paper is based on a part of a dissertation submitted to the Brooklyn Polytechnic Institute in partial fulfillment of the requirements for the degree of Doctor of Philosophy (June 1972).

PROOF: After justifying termwise differentiation, which follows directly from $|t| < 1$, and $|a_j| < 1$, the proof of the lemma is a straightforward application of the evaluation:

$$P_n(k) = (1/n!) \cdot \partial^n G(k, t)/\partial t^n|_{t=0}. \tag{3}$$

The main theorem of this paper is the following:

THEOREM: Let $P_n(k)$ denote the probability that in n consecutive trials we will have k successes, given that the probability of a success on any given trial is p_s, $0 < p_s < 1$, where s is the number of previous successes up to this particular trial (the values $p_s = 0$, 1 are purposefully excluded as uninteresting). We take $p_i \neq p_j$ for $i \neq j$ and let $q_s \equiv 1 - p_s$. Then, for $n = 1, 2, 3, \ldots$, we have:

$$P_n(k) = \begin{cases} q_0^n & \text{for } k = 0 \\ \prod_{i=0}^{k-1} p_i \times \sum_{j=0}^{k} \left[(q_j)^n \cdot \prod_{\substack{s=0 \\ s \neq j}}^{k} (q_j - q_s)^{-1} \right], & \text{for } 1 \leqslant k \leqslant n \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and these are normalized in that $\sum_{k=0}^{\infty} P_n(k) = 1$.

PROOF: The following difference equation can be constructed directly from the statements in the theorem:

$$P_n(k) = p_{k-1} \cdot P_{n-1}(k-1) + q_k \cdot P_{n-1}(k), \tag{5}$$

for $k = 1, 2, \ldots, n$. Also we have:

$$P_n(0) = q_0^n. \tag{6}$$

What (5) says is simply that in order to have k successes in n consecutive trials we must either have $k-1$ successes in $n-1$ trials and then a success on the nth trial, or else k successes in $n-1$ trials and then no success on the nth trial. What (6) says is that in order to have no successes in n trials we must have n consecutive failures, where q_0 is the probability of failure at each trial as there would be no previous successes at any trial.

The boundary conditions on this difference equation are:

$$P_n(k) = 0, \text{ for } k > n, \text{ or } k < 0, \tag{7}$$

$$P_0(0) = 1. \tag{8}$$

We introduce the generating function $G(k, t)$, with $|t| < 1$, defined as follows:

$$G(k, t) \equiv \sum_{n=0}^{\infty} P_n(k) \cdot t^n = \sum_{n=k}^{\infty} P_n(k) \cdot t^n, \tag{9}$$

where the second form follows from the condition (7) for $k > n$. As starting condition on this generating function we have:

$$G(0, t) = \sum_{n=0}^{\infty} P_n(0) \cdot t^n = \sum_{n=0}^{\infty} (q_0 t)^n = (1 - q_0 t)^{-1}, \tag{10}$$

where use is made of the fact that $|q_0 t| < 1$, as well as (6).

Upon multiplying both sides of (5) by t^n and then summing over all values of n we produce the following result:

$$\sum_{n=k}^{\infty} P_n(k) \cdot t^n = p_{k-1} \cdot \sum_{n=k}^{\infty} P_{n-1}(k-1) \cdot t^n + q_k \cdot \sum_{n=k}^{\infty} P_{n-1}(k) \cdot t^n$$

$$= t p_{k-1} \cdot \sum_{n=k}^{\infty} P_{n-1}(k-1) \cdot t^{n-1} + t q_k \cdot \sum_{n=k}^{\infty} P_n(k) \cdot t^n. \tag{11}$$

Here we used the fact that $P_{k-1}(k) = 0$.

A statement in terms of the generating function which is equivalent to (11) is

$$G(k, t) = t p_{k-1} \cdot G(k-1, t) + t q_k \cdot G(k, t), \tag{12}$$

which can be rewritten as

$$G(k, t) = (t p_{k-1} / 1 - q_k t) \cdot G(k-1, t). \tag{13}$$

From (13), by induction, and by making use of (10), one finds that

$$G(k, t) = \frac{t^k p_0 p_1 \ldots p_{k-1}}{(1 - q_0 t)(1 - q_1 t) \ldots (1 - q_k t)}. \tag{14}$$

This last expression for $G(k, t)$ can be expended by partial fractions with the following result:

$$G(k, t) = t^k \cdot \prod_{i=0}^{k-1} p_i \left\{ \frac{q_0^k}{(1 - q_0 t)(q_0 - q_1)(q_0 - q_2) \ldots (q_0 - q_k)} \right. \tag{15}$$

$$+ \frac{q_1^k}{(1 - q_1 t)(q_1 - q_0)(q_1 - q_2) \ldots (q_1 - q_k)}$$

$$\left. + \ldots + \frac{q_k^k}{(1 - q_k t)(q_k - q_0)(q_k - q_1) \ldots (q_k - q_{k-1})} \right\}$$

$$= \sum_{j=0}^{k} \left(\prod_{i=0}^{k-1} p_i \times \left[\prod_{\substack{s=0 \\ s \neq j}}^{k} (q_j - q_s)^{-1} \right] \cdot [(t q_j)^k / (1 - q_j t)] \right).$$

At this point we make use of the Lemma, where we identify a_j with q_j and $g_j(k)$, $j=0, 1, 2, \ldots, k$, with the terms

$$\prod_{i=0}^{k-1} p_i \cdot \prod_{\substack{s=0 \\ s\neq j}}^{k} (q_j - q_s)^{-1}, \qquad j=0, 1, 2, \ldots, k, \tag{16}$$

as found in (15). From the Lemma we conclude that

$$P_n(k) = \prod_{i=0}^{k-1} p_i \times \sum_{j=0}^{k} (q_j)^n \cdot \prod_{\substack{s=0 \\ s\neq j}}^{k} (q_j - q_s)^{-1}, \qquad k=1, 2, \ldots, n \tag{17}$$

as stated in the Theorem. This, along with (6) and (7), completes the first part of the theorem. Now we must show that $\sum_{k=0}^{\infty} P_n(k) = 1$, or, due to (7), that $\sum_{k=0}^{n} P_n(k) = 1$. While there are various ways in which this can be accomplished, a rather interesting way is the following.

We introduce the polynomial $U_n(x)$, defined as

$$U_n(x) \equiv a_0 + \sum_{j=1}^{n} a_j \cdot \prod_{k=0}^{j-1} (x - q_k). \tag{18}$$

If we determine the values of the coefficients a_k for which $U_n(q_i) = q_i^n$, for $i=0, 1, \ldots, n$, we find the values

$$a_k = \sum_{j=0}^{k} (q_j)^n \cdot \prod_{\substack{s=0 \\ s\neq j}}^{k} (q_j - q_s)^{-1}, \qquad \text{for } k=1, 2, \ldots, n \tag{19}$$

$$a_0 = (q_0)^n, \qquad \text{for } k=0.$$

These coefficients are *Newton's divided differences* of the function $y=x^n$.

Substituting the evaluations of (19) back into (18) we have as the result

$$U_n(x) = q_0^n + \sum_{k=1}^{n} \left[\prod_{i=0}^{k-1} (x - q_i) \cdot \sum_{j=0}^{k} (q_j)^n \cdot \prod_{\substack{s=0 \\ s\neq j}}^{k} (q_j - q_s)^{-1} \right], \tag{20}$$

which means that with $x=1$, recalling (17), we have $U_n(1) = \sum_{k=0}^{n} P_n(k)$. However, as $U_n(x)$ and x^n assume the same values at $n+1$ distinct points then $U_n(x) \equiv x^n$: being both of degree n with their coefficients over the field of real numbers [2]. Hence, $U_n(1) = (1)^n = 1$, thus completing the proof.

EVALUATING $Pr[D_n = k/R = r]$

As indicated in the introduction we will produce the new evaluation for $Pr[D_n = k/R = r]$ by specifying p_s as found in the results of the previous section.

If we have a particle just starting its crossing and if there have been s previous particles absorbed, with initially r absorption points on the common path, then there are $r-s$ points left for this particle to contact and so the probability of making it across without being absorbed is q^{r-s}. It follows that $1-q^{r-s}$ is the probability that this particle will be absorbed at some point on the path.

To identify our particular absorption problem with the generalized Bernoulli scheme, thus allowing us to make use of the general *pdf* $P_n(k)$, given by (4), we simply have to make these specifications:

$$P_s = 1-q^{r-s}, \qquad q_s = q^{r-s}, \qquad s = 0, 1, 2, \ldots, \min(n, r). \tag{21}$$

In most practical situations R would be a random variable, and so we would not know beforehand the value of r. This means that as our specifications involve a conditioning of R we must interpret $P_n(k)$ as a conditional distribution: which is of course $Pr[D_n = k/R = r]$.

Making the substitution of (21) into (4), and simplifying the results as needed, we get from the previous theorem the following:

THEOREM: The conditional probability distribution function of having D_n absorptions in n particle crossings of a path that had initially R absorption points on it is, for $n = 1, 2, \ldots$, given by

$$Pr[D_n = k/R = r] = \begin{cases} q^{nr} & \text{for } k = 0 \\ \prod_{i=0}^{k-1} (1-q^{r-i}) \cdot q^{r(n-k)} \cdot F(n, k; q), & \text{for } 1 \leqslant k \leqslant \min(n, r) \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where

$$F(n, k; q) \equiv \sum_{j=0}^{k} q^{-nj} \cdot \prod_{\substack{s=0 \\ s \neq j}}^{k} (q^{-j} - q^{-s})^{-1}. \tag{23}$$

The introduction of $F(n, k; q)$ is notationally convenient, but also the symmetry property $F(n, k; q) = F(n, n-k; q)$ for $1 \leqslant k \leqslant n$, $n > 0$ and $n \geqslant k$, affords a way by which the computations needed in evaluating (22) can be reduced. The proof of this property is lengthy and can be found in [3].

Some values of $F(n, k; q)$ are presented in Table 1, where the cited symmetry property is clearly evident. In Table 2 we list values of $Pr[D_n = k/R = r]$ over a restricted range of values of the parameters involved.

THE UNCONDITIONAL DISTRIBUTION $Pr[D_n = k]$

Here we seek an expression for the unconditional probability distribution $Pr[D_n = k]$ that in n particle crossing attempts there will be exactly k absorptions. If we are given the *pdf* covering the number of absorption points on the path, i.e., we know $Pr[R = r]$, $r = 0, 1, 2, \ldots$, then formally we have

$$Pr[D_n = k] = \sum_{r=0}^{\infty} Pr[D_n = k/R = r] \cdot Pr[R = r]. \tag{24}$$

TABLE 1. *Values of* F(n, k; q)

		Values of q			
n	k	0.10	0.20	0.30	0.40
1	0	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
	1	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
2	0	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
	1	0.110000000000ex 02	0.600000000000ex 01	0.433333333329ex 01	0.350000000000ex 01
	2	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
3	0	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
	1	0.111000000000ex 03	0.310000000000ex 02	0.154444444443ex 02	0.975000000000ex 01
	2	0 111000000000ex 03	0.310000000000ex 02	0.154444444443ex 02	0.975000000000ex 01
	3	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
4	0	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
	1	0.111100000000ex 04	0.156000000000ex 03	0.524814814805ex 02	0.253750000000ex 02
	2	0.112110000000ex 05	0.806000000000ex 03	0.187049382711ex 03	0.706875000000ex 02
	3	0.111100000000ex 04	0.156000000000ex 03	0.524814814805ex 02	0.253750000000ex 02
	4	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
5	0	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
	1	0.111110000000ex 05	0.781000000000ex 03	0.175938271601ex 03	0.644375000000ex 02
	2	0.112221100000ex 07	0.203060000000ex 05	0.213080795604ex 04	0.467171875000ex 03
	3	0.112221100000ex 07	0.203060000000ex 05	0.213080795604ex 04	0.467171875000ex 03
	4	0.111110000000ex 05	0.781000000000ex 03	0.175938271601ex 03	0.644375000000ex 02
	5	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01

At this point (22) could be used directly in (24), but a more compact result is obtained if we first make use of the following expression in (22) (see [3] for the proof of this):

$$\prod_{i=0}^{k-1} (1 - q^{r-i}) = 1 + \sum_{m=1}^{k} q^{m(r-k)} \cdot F(m-k-1, m; q). \tag{25}$$

It is to be noted here that the first index in $F(m-k-1, m; q)$ —i.e., the quantity $m-k-1$—will be negative over the range of summation. While this would not be meaningful under the former interpretation of this index (as the number of crossing attempts, n), it in no way violates the actual functional definition of F as given by (23).

Now, by using (25) in (22), and in turn using this in (24), then with the definition

$$E_R[q^{bR}] \equiv \sum_{r=0}^{\infty} q^{bR} \cdot Pr[R=r], \qquad \text{for any constant } b, \tag{27}$$

we will produce the following theorem.

THEOREM: The unconditional probability distribution function $Pr[D_n=k]$ for having D_n absorptions in n particle crossing attempts, $n=1, 2, \ldots$, is given by

as specified by (23).

0.50	0.60	0.70	0.80	0.90
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.300000000000ex 01	0.266666666640ex 01	0.242857142855ex 01	0.225000000000ex 01	0.211111111110ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.700000000000ex 01	0.544444444437ex 01	0.446938775505ex 01	0.381250000000ex 01	0.334567901230ex 01
0.700000000000ex 01	0.544444444437ex 01	0.446938775505ex 01	0.381250000000ex 01	0.334567901230ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.150000000000ex 02	0.100740740739ex 02	0.738483965006ex 01	0.576562500000ex 01	0.471742112470ex 01
0.350000000000ex 02	0.205679012343ex 02	0.135905872552ex 02	0.976953124960ex 01	0.747614692880ex 01
0.150000000000ex 02	0.100740740739ex 02	0.738483965006ex 01	0.576562500000ex 01	0.471742112470ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01
0.310000000000ex 02	0.177901234564ex 02	0.115497709286ex 02	0.820703125000ex 01	0.624157902750ex 01
0.155000000000ex 03	0.672071330570ex 02	0.351207320074ex 02	0.210305175775ex 02	0.139472321471ex 02
0.155000000000ex 03	0.672071330570ex 02	0.351207320074ex 02	0.210305175775ex 02	0.139472321471ex 02
0.310000000000ex 02	0.177901234564ex 02	0.115497709286ex 02	0.820703125000ex 01	0.624157902750ex 01
0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01	0.100000000000ex 01

28) $$Pr[D_n=k] \begin{cases} E_R[q^{nR}] & \text{for } k=0 \\ F(n,\, k;\, q)\cdot E_R[q^{(n-k)R}]+\sum_{m=1}^{k} q^{-mk}F(m-k-1,\, m;\, q)\cdot E_R[q^{(n-k+m)R}] & \text{for } 1\leqslant k\leqslant n \\ 0 & \text{otherwise.} \end{cases}$$

It is to be noted that $E_R[q^{bR}]$, as defined above, is simply the probability generating function of the random variable R evaluated at the point $t=q^b$. Thus, for example, with R distributed in accordance with the Poisson distribution, with intensity λ, we would have

29) $$Pr[R=r]=(\lambda^r/r!)\cdot\exp(-\lambda),$$

and

30) $$E_R[q^{bR}]=\exp\{-\lambda(1-q^b)\}\cdot$$

From (28) we have that the probability of no absorptions in n attempts, and the probability of at least one absorption in n attempts are given, respectively, by $E_R[q^{nR}]$ and $1-E_R[q^{nR}]$. For the case with R

TABLE 2. *Values of* $\Pr[D_n = k/R = r]$

	$r=1,\ q=0.10,\ n=1$	$r=1,\ q=0.20,\ n=1$	$r=1,\ q=0.30,\ n=1$	$r=1,\ q=0.40,\ n=1$
$k=0$	0.100000000000ex−00	0.200000000000ex 00	0.299999999999ex 00	0.400000000000ex 00
$k=1$	0.900000000000ex 00	0.800000000000ex 00	0.700000000010ex 00	0.600000000000ex 00
	$r=1,\ q=0.10,\ n=2$	$r=1,\ q=0.20,\ n=2$	$r=1,\ q=0.30,\ n=2$	$r=1,\ q=0.40,\ n=2$
$k=0$	0.100000000000ex−01	0.400000000001ex−01	0.900000000000ex−01	0.160000000000ex 00
$k=1$	0.990000000000ex 00	0.960000000000ex 00	0.909999999938ex 00	0.840000000000ex 00
$k=2$	0.000000000000ex 00	0.000000000000ex 00	0.000000000000ex 00	0.000000000000ex 00
	$r=5,\ q=0.10,\ n=1$	$r=5,\ q=0.20,\ n=1$	$r=5,\ q=0.30,\ n=1$	$r=5,\ q=0.40,\ n=1$
$k=0$	0.100000000000ex−04	0.320000000000ex 03	0.243000000001ex−02	0.102400000000ex−01
$k=1$	0.999990000000ex 00	0.999680000000ex 00	0.997570000000ex 00	0.989760000000ex 03
	$r=5,\ q=0.10,\ n=2$	$r=5,\ q=0.20,\ n=2$	$r=5,\ q=0.30,\ n=2$	$r=5,\ q=0.40,\ n=2$
$k=0$	0.100000000000ex−09	0.102400000000ex−06	0.590489999990ex−05	0.104857600000ex−03
$k=1$	0.109998900000ex−03	0.191938560000ex−02	0.105044120999ex−01	0.354729984000ex−01
$k=2$	0.999890001000ex 00	0.998080512000ex 00	0.989489683000ex 00	0.964422144000ex 00
	$r=5,\ q=0.10,\ n=3$	$r=5,\ q=0.20,\ n=3$	$r=5,\ q=0.30,\ n=3$	$r=5,\ q=0.40,\ n=3$
$k=0$	0.100000000000ex−14	0.327680000030ex−10	0.143489069998ex−07	0.107374182400ex−05
$k=1$	0.110998890000ex−07	0.317338419200ex−05	0.909762891011ex−04	0.101189261721ex−02
$k=2$	0.110987790111ex−02	0.990095867904ex−02	0.371355478019ex−01	0.962879068569ex−01
$k=3$	0.998890110999ex 00	0.990095867904ex 00	0.962773461559ex 00	0.902699126784ex 00
	$r=5,\ q=0.10,\ n=4$	$r=5,\ q=0.20,\ n=4$	$r=5,\ q=0.30,\ n=4$	$r=5,\ q=0.40,\ n=4$
$k=0$	0.100000000000ex−19	0.104857600004ex−13	0.348678440105ex−10	0.109951162777ex−07
$k=1$	0.111098889000ex−11	0.511017222189ex−08	0.751221980861ex−06	0.269671977083ex−04
$k=2$	0.112097668012ex−05	0.823759762089ex−04	0.109289917179ex−02	0.714841420501ex−02
$k=3$	0.110976691331ex−01	0.494255857257ex−01	0.122782499549ex−00	0.234557341103ex−00
$k=4$	0.988901209889ex 00	0.950492033187ex 00	0.876123850018ex 00	0.758267266498ex 00
	$r=5,\ q=0.10,\ n=5$	$r=5,\ q=0.20,\ n=5$	$r=5,\ q=0.30,\ n=5$	$r=5,\ q=0.40,\ n=5$
$k=0$	0.100000000000ex−24	0.335544319999ex−17	0.847288609462ex−13	0.112589990684ex−09
$k=1$	0.111108888900ex−15	0.818675795909ex−11	0.611968116030ex−08	0.701242787618ex−06
$k=2$	0.112208755791ex−08	0.664109805681ex−06	0.302534147209ex−04	0.483775346568ex−03
$k=3$	0.112096547035ex−03	0.205874039740ex−02	0.121138158533ex−01	0.442200902721ex−01
$k=4$	0.109876813430ex−00	0.237546968933ex−00	0.374569229587ex 00	0.500335073125ex−00
$k=5$	0.890011088900ex 00	0.760393626549ex 00	0.613286695021ex 00	0.454960359898ex 00

distributed as the Poisson these become

(31) $$Pr[D_n = 0] = \exp\{-\lambda(1-q^n)\}$$

and

(32) $$Pr[D_n \geq 1] = 1 - \exp\{-\lambda(1-q^n)\}$$

as specified by (22).

$r=1, q=0.50, n=1$	$r=1, q=0.60, n=1$	$r=1, q=0.70, n=1$	$r=1, q=0.80, n=1$	$r=1, q=0.90, n=1$
0.500000000000ex 00	0.599999999998ex 00	0.700000000000ex 00	0.800000000000ex 00	0.900000000000ex 00
0.500000000000ex 00	0.400000000010ex 00	0.300000000000ex 00	0.200000000000ex 00	0.100000000000ex 00
$r=1, q=0.50, n=2$	$r=1, q=0.60, n=2$	$r=1, q=0.70, n=2$	$r=1, q=0.80, n=2$	$r=1, q=0.90, n=2$
0.250000000000ex 00	0.359999999998ex 00	0.490000000000ex 00	0.640000000000ex 00	0.809999999996ex 00
0.750000000000ex 00	0.640000000006ex 00	0.509999999995ex 00	0.360000000000ex 00	0.189999999999ex−00
0.000000000000ex 00	0.000000000000ex 00	0.000000000000ex 00	0.000000000000ex 00	0.000000000000ex 00
$r=5, q=0.50, n=1$	$r=5, q=0.60, n=1$	$r=5, q=0.70, n=1$	$r=5, q=0.80, n=1$	$r=5, q=0.90, n=1$
0.312500000000ex−01	0.777600000001ex−01	0.168069999999ex 00	0.327680000000ex 00	0.590489999997ex 00
0.968750000000ex 00	0.922240000000ex 00	0.831930000010ex 00	0.672320000000ex 00	0.409510000010ex 00
$r=5, q=0.50, n=2$	$r=5, q=0.60, n=2$	$r=5, q=0.70, n=2$	$r=5, q=0.80, n=2$	$r=5, q=0.90, n=2$
0.976562500000ex−03	0.604661759989ex−02	0.282475248998ex−01	0.107374182400ex 00	0.348678440097ex 00
0.908203125000ex−01	0.191235686398ex−00	0.339568868138ex 00	0.495688089600ex 00	0.510491070905ex 00
0.908203125000ex 00	0.802717696009ex 00	0.632183607015ex 00	0.396937728000ex 00	0.140830489007ex 00
$r=5, q=0.50, n=3$	$r=5, q=0.60, n=3$	$r=5, q=0.70, n=3$	$r=5, q=0.80, n=3$	$r=5, q=0.90, n=3$
0.305175781250ex−04	0.470184984571ex−03	0.474756150989ex−02	0.351843720889ex−01	0.205891132092ex 00
0.662231445312ex−02	0.303605775718ex−01	0.105030448619ex−00	0.275223651811ex−00	0.477720499621ex 00
0.198669433593ex−00	0.339838563777ex−00	0.474877360066ex 00	0.495886364835ex 00	0.278223305772ex−00
0.794677734375ex 00	0.629330673679ex 00	0.415344629815ex 00	0.193705611264ex 00	0.381650625223ex−01
$r=5, q=0.50, n=4$	$r=5, q=0.60, n=4$	$r=5, q=0.70, n=4$	$r=5, q=0.80, n=4$	$r=5, q=0.90, n=4$
0.953674316406ex−06	0.365615843995ex−04	0.797922662969ex−03	0.115292150461ex−01	0.121576654588ex 00
0.443458557127ex−03	0.436835425348ex−02	0.291674495605ex−01	0.136386764824ex−00	0.397746897300ex−00
0.310420989989ex−01	0.998309764935ex−01	0.242695572411ex 00	0.416385862807ex−00	0.367112869735ex−00
0.372505187986ex 00	0.492992476523ex 00	0.515513294171ex−00	0.365964137248ex−00	0.106312216508ex−00
0.596008300781ex 00	0.402771631160ex 00	0.211825761205ex 00	0.697340200550ex−01	0.217540856388ex−01
$r=5, q=0.50, n=5$	$r=5, q=0.60, n=5$	$r=5, q=0.70, n=5$	$r=5, q=0.80, n=5$	$r=5, q=0.90, n=5$
0.298023223876ex−07	0.284302880292ex−05	0.134106861964ex−03	0.377789318630ex−02	0.717897987685ex−01
0.286400318145ex−04	0.599857266833ex−03	0.766692044046ex−02	0.636153407321ex−01	0.310748595140ex−00
0.429600477216ex−02	0.253657064646ex−01	0.105408926258ex−00	0.293712307709ex 00	0.404410439999ex 00
0.120288133620ex−00	0.255744777113ex−00	0.412052505214ex 00	0.437413348883ex 00	0.185600483061ex 00
0.577383041378ex 00	0.557178163669ex 00	0.411189812864ex 00	0.187534305453ex−00	0.267255468293ex−01
0.298004150390ex 00	0.161108652468ex 00	0.635477283615ex−01	0.139468040110ex−01	0.725136187961ex−03

Since $q\epsilon(0, 1)$, making $1-q^n \leqslant n\cdot(1-q)$, it follows from the above that for the Poisson distributed R,

$$Pr[D_n=0] \geqslant \exp(-\lambda np) \tag{33}$$

and

$$Pr[D_n \geqslant 1] \leqslant 1-\exp(-\lambda np). \tag{34}$$

Furthermore, still under the Poisson distributed R assumption, we obtain from (28), using (30), the evaluation

$$(35) \qquad Pr[D_n=k]=F(n,k;q)\cdot\left\{e^{-\lambda(1-q^{n-k})}+\sum_{m=1}^{k} q^{-mk}\cdot F(m-k-1,m;q)\cdot e^{-\lambda(1-q^{n-k+m})}\right\}.$$

SURVIVAL PROBABILITIES

In certain applications of this type of absorption problem the event of main interest is the number of particles which survive the crossing rather than the number absorbed. For example, this would be the case if our particles were military personnel (and/or equipment) and the absorption points were land mines in a mine field. Here we are interested in the event $S(n+1)$ that the $(n+1)-st$ particle will survive its attempt at crossing the path.

The probability of the survival of any particle is conditioned by the number of absorption points still active at the time this particular particle starts to cross over the path. Thus, if this is the $(n+1)-st$ particle its survival depends directly on the difference between R and D_n. In fact we have that

$$(36) \qquad Pr[S(n+1)/D_n,R]=q^{R-Dn}, \qquad \text{for } 0\leqslant D_n\leqslant \min(n,R).$$

Formally we have the following evaluation

$$(37) \qquad Pr[S(n+1)]=\sum_{k=o}^{n}\sum_{r=0}^{\infty} Pr[S(n+1)/D_n=k,R=r]\cdot Pr[D_n=k/R=r]\cdot Pr[R=r].$$

Thus, by introducing (36) and (22) into (37) and making use of the previous definition of $E_R[q^{bR}]$, we produce this result:

THEOREM: The probability distribution function for the probability that the $(n+1)-st$ particle to attempt to cross a path (with initially R absorption points on it) will survive the crossing is $Pr[S(n+1)]$ where:

$$(37) \quad Pr[S(n+1)]=\sum_{k=0}^{n} q^{-k}\cdot F(n,k;q) \cdot\left(E_r[q^{(n-k+1)R}]+\sum_{m=1}^{k} q^{-mk}\cdot F(m-k-1,m;q)\cdot E_R[q^{(m+n-k+1)R}]\right).$$

In particular, the probability of the second particle surviving, regardless of what the first particle did, is

$$(38) \quad Pr[S(2)]=\sum_{k=0}^{1} q^{-k}F(1,k;q)\cdot\left(E_R[q^{(2-k)R}]+\sum_{m=1}^{k} q^{-mk}\cdot F(m-k-1,m;q)\cdot E_R[q^{(2-k+m)R}]\right)$$

$$(39) \qquad =E_R[q^R]\cdot q^{-1}\cdot F(1,1;q)+E_R[q^{2R}]\cdot F(1,0;q)+q^{-2}\cdot F(-1,1;q)\cdot F(1,1;q),$$

and since $F(1,1;q)=F(1,0;q)=1$, and $F(-1,1;q)=-q$, it follows from this expression that

$$(40) \qquad Pr[S(2)]=(1/q)\cdot E_R[q^R]-(p/q)\cdot E_R[q^{2R}].$$

In particular, when R is distributed as the Poisson with intensity λ, then by using (30) as needed in (40) we have this evaluation:

$$Pr[S(2)] = (1/q) \cdot \exp\{-\lambda(1-q)\} - (p/q) \cdot \exp\{-\lambda(1-q)\}. \tag{41}$$

In a similar manner the new expression for $Pr[D_n = k/R = r]$ can be used in the other conditional survival distributions developed in [1] as well as in the case where two paths are considered.

REFERENCES

[1] Zacks, S. and D. Goldfarb, "Survival Probabilities in crossing a Field Containing Absorption Points," Nav. Res. Log. Quart. 13, 35–48 (1966).

[2] Birkoff, G., and S. Mclane, *A Survey of Modern Algebra* (Macmillian, N.Y., 1958).

[3] Parsons, J., "Absorption Probabilities in Moving Through a Spatial Structure Containing Adsorption Sites," unpublished Ph.D. dissertation, Brooklyn Polytechnic Institute (1972).

OPTIMAL AND SUBOPTIMAL PROCEDURES IN GROUP SEQUENTIAL SAMPLING

M. Spahn

The Mitre Corporation

and

S. Ehrenfeld*

Baruch College, City University of New York (CUNY)

ABSTRACT

This paper investigates the problem of choosing between two simple hypothesis, H_0 and H_1, in terms of independent, identically distributed random variables, when observations can be taken in groups. At any stage in the decision process it must be decided whether to stop and take action now or to continue, in which case the size of the next group of observations must be decided upon. The problem is to find an optimal procedure incorporating a stopping, group size (batch) and terminal action rule.

It is proven, in general, that the optimal stopping and terminal action rule is of the sequential probability ratio type (SPRT). Fixed stopping rules of the SPRT type are studied and an iterative procedure of the policy improvement type, both with and without a value determination step, is developed. It is shown, for the general situation, that both the average risk and scheduling rule converge to the optima. Also, six suboptimal scheduling rules are considered with respect to the average risks they achieve. Numerical results are presented to illustrate the effectiveness of the procedures.

1. INTRODUCTION

In this paper we consider the problem of testing two simple hypotheses, H_0: $\theta=\theta_0$ vs H_1: $\theta=\theta_1$, by sequentially observing independent, identically distributed random variables, when the cost of sampling is such that observations are best taken in groups. When the cost of sampling is simply a constant cost, c, per observation, then Wald's Sequential Probability Ratio Test (SPRT) is well known to have optimal properties for this simple hypothesis test (see [3], [4], [8], [9], [11]), which, in turn, stems from the fact that it is a Bayes procedure with respect to some prior distribution; $\pi = Pr(H_0)$, $1-\pi = Pr(H_1)$. That is, for some π, the SPRT is the procedure, δ, which minimizes the expected risk,

$$R_\delta(\pi) = \pi(E_0(\text{cost}) + \omega_0\alpha) + (1-\pi)(E_1(\text{cost}) + \omega_1\beta), \tag{1}$$

where ω_i is the loss of choosing H_{1-i} when the true state of nature is $\theta=\theta_i$, $i=0, 1$: $\alpha = Pr$ (accepting $H_1 \mid \theta=\theta_0$), $\beta = Pr$ (accepting $H_0 \mid \theta=\theta_1$); and E_i (cost) is the expected sampling cost when $\theta=\theta_i$, $i=0, 1$.

*Research was supported by the National Science Foundation under Grant GK-14073.

The SPRT as a sequential testing procedure consists of two parts, a stopping rule and a terminal decision rule; it involves no scheduling rule. That is, if at any stage the decision to continue testing has been made, exactly one additional observation will be made available for the decisions of the next stage. That the SPRT is a Bayes procedure and has no scheduling or batching rule is due to the linear cost structure, ck, of the problem for which it is optimal. Under such a sampling cost structure there is no reason for, or penalty for not, scheduling more than one observation at any stage.

The problem investigated here is finding Bayes procedures, which minimize $R_\delta(\pi)$, for sequentially testing two simple hypotheses, H_0 vs H_1, when the cost, at any stage, of scheduling k observations is $c(k)$ such that $c(k)$ increases monotonically beyond bound with increasing k. Such procedures must, of necessity, consist of three parts; a stopping rule, a terminal decision rule, and a scheduling rule. That is, at each stage 1) a decision must be made to continue or stop testing; 2) if the decision to stop has been made, a choice must be made between H_0 and H_1, 3) if the decision to continue has been made, it must further be decided how many additional observations should be scheduled for the next stage.

One example of such a sampling cost structure is $c(k) = d + ck$ at each stage which may be interpreted as a set-up cost, d, constant and independent of the number (≥ 1) of scheduled observations, plus a constant cost, c, per observation scheduled. Its significance lies in the fact that it is a more accurate model of the costs of many testing situations than the model of ck, and contains the ck model as a special case ($d = 0$). One particular situation, among others more obvious, which may be so modeled is one in which the experiments yielding the observations are very time consuming, and time is of value. Scheduling one or $k \geq 1$ experiments requires the same amount of valuable time.

The results of Osaki and Mine [5], and Ross [6] show that there exists an optimum stationary procedure. Ehrenfeld [2] considers the special case of the log-likelihood ratio being a finite number of integral multiples of a given constant. In that case he uses the results of Derman [1] to arrive at a linear programming formulation which yields the optimum risks and scheduling rule.

In section 2 we show that the optimum stopping and terminal decision rule is of the SPRT type, and develop a recursive formulation of the problem. The probability structure of the problem is the same as that of the SPRT, and so for fixed stopping rules of the SPRT type many of the important properties of the SPRT carry directly over, essentially unaffected by any scheduling rule. For example, the testing procedure must stop with probability one, and the expected numbers of observations and of stages are both finite. As in the SPRT, the information available at any stage can be summarized by the current posterior probability, π_n, of H_0 given observations $x_1, x_2, \ldots, x_n$. Thus, in view of the results of Ross [6], if we fix the stopping rule, we can search for stationary scheduling rules as functions of π_n, or equivalently for given a-priori π, of the long-likelihood ratio. Furthermore, since the average risk is finite, arguments similar to those used by Derman [1] and Ross [6] show that randomization in the scheduling rule is not necessary.

In sections 3 through 7 we consider fixed stopping rules of the SPRT type and develop an iterative procedure of the policy improvement type, both with and without a value determination step, similar to that presented by Ehrenfeld [2] in the discrete case. We prove, for the general situation, discrete or continuous log-likelihood ratio, that both the average risk and scheduling rule converge to the optimum. We also consider six suboptimal scheduling rules based upon the first few steps of the iterative method, and compare them with respect to the average risks they achieve. We give numerical results in the case of the symmetrical binomial test ($p_1 = 1 - p_0$) for which the computational problem becomes discrete.

For the cases examined the iterative procedure converges within the first few steps, and several of the suboptimal scheduling rules are seen to be extremely good, if not optimal.

2. Notation and General Considerations

Let $P_{in} = \prod_{j=1}^{n} f_i(x_j)$, $i = 0, 1$ where f_i, $i = 0, 1$, is the density or probability function (depending upon whether the X_j are continuous or discrete) of each of the independent, identically distributed X_j, given that $\theta = \theta_i$. Then the log-likelihood ratio is $\nu_n = \ln \frac{P_{1n}}{P_{0n}} = \sum_{j=1}^{n} z_j$, where $z_j = \ln \left(\frac{f_1(x_j)}{f_0(x_j)} \right)$.

A procedure, δ, for testing $H_0 : \theta = \theta_0$ vs $H_1 : \theta = \theta_1$ consists of 1) a terminal decision rule, 2) a stopping rule if δ is sequential, and, again if δ is sequential, 3) a scheduling rule. The losses due to incorrect terminal decisions are w_0 = the loss of accepting H_1 when H_0 is true and w_1 = the loss of accepting H_0 when H_1 is true.

Associated with each procedure, δ, are $\alpha = \alpha(\delta) = Pr(\text{accept } H_1 | \theta = \theta_0)$ and $\beta = \beta(\delta) = Pr(\text{accept } H_0 | \theta = \theta_1)$. The a-priori probability that $\theta = \theta_0 (H_0 \text{ is true})$ is π and the posterior probability that $\theta = \theta_0$ after observations $x_1, \ldots, x_n$ have been observed is

$$\pi(x_1, \ldots, x_n) = \pi_n = \frac{1}{1 + \left(\frac{1 - \pi}{\pi} \right) e^{v_n}} \tag{2}$$

For any procedure, δ, the Bayes risk is given by (1).

We wish to find procedures, δ, which minimize $R_\delta(\pi)$. Without having to say anything about the scheduling rule, we can characterize the stopping and terminal decision rules of such optimal procedures, δ. The following theorem does just that.

THEOREM 1: Let $\rho(\pi) = \inf_{\delta \in C} R_\delta(\pi)$ where C is the class of all procedures requiring at least 1 observation. Let δ_i, for $i = 0, 1$, be the procedure which rejects H_i without any observations. Let π' and π'' satisfy:

$$R_{\delta_0}(\pi') = \rho(\pi'); \; R_{\delta_1}(\pi'') = \rho(\pi''), \tag{3}$$

if

$$\rho \left(\frac{w_1}{w_0 + w_1} \right) < \frac{w_0 w_1}{w_0 + w_1}$$

otherwise

$$\pi' = \pi'' = \frac{w_1}{w_0 + w_1}. \tag{4}$$

If $0 < \pi' < \pi'' < 1$ then for all $\pi' \leq \pi \leq \pi''$ an optimal stopping and terminal decision rule for minimizing $R_\delta(\pi)$ is an SPRT rule with boundaries on the likelihood ratio axis

$$B = \left(\frac{\pi}{1 - \pi} \right) \left(\frac{1 - \pi''}{\pi''} \right) < A = \left(\frac{\pi}{1 - \pi} \right) \left(\frac{1 - \pi'}{\pi'} \right). \tag{5}$$

PROOF: The proof parallels the proof of Lemma 5 in Lehmann [4], Section 3.12, establishing the concavity and continuity of $\rho(\pi)$ on the interval [0, 1], and observing that at any stage the cost of past observations is irretrievable, with the situation exactly the same as at the first step, π being replaced by π_n as given in (2).

Thus if v is the log-likelihood ratio $= \ln \frac{P_{1n}}{P_{0n}} = \sum_{j=1}^{n} z_j$ $(z_0 = 0)$ then there are two constants $b = \ln B < 0$, and $a = \ln A > 0$ such that

if $v \leqslant b$, stop and accept H_0

if $b < v < a$, continue sampling

if $a \leqslant v$, stop and accept H_1

We now fix the stopping and terminal decision rules to be of this form and consider finding associated optimal and suboptimal scheduling rules. The stopping constants, a and b, may be assumed to be given, or chosen to achieve given α and β via the Wald approximations

$$b = \ln\left(\frac{\beta}{1-\alpha}\right), \quad a = \ln\left(\frac{1-\beta}{\alpha}\right).$$

We also fix the a-priori probability of H_0 and relabel it π_0, the subscript distinguishing it from the posterior probability π_n. In what follows we represent a scheduling rule as $k(v)$, a function of the log-likelihood ratio, v. Thus whether to continue, and if so, how many observations to schedule, $k(v)$, depends solely upon the position, v, on the log-likelihood axis, the starting position being $v = 0$. The validity of this approach stems from the one to one relationship, (2), that exists between the log-likelihood ratio and π_n, and from the fact that all the information at any stage is contained in π_n.

Specifying a scheduling rule, $k(v)$, now completely determines a testing procedure, δ. Let $L_\delta(v)$ be the average loss or risk starting at v until termination. Observe that $L_\delta(0) = R_\delta(\pi_0)$. Also if $v \leqslant b$ we stop and accept H_0 with loss

$$L_\delta(v) = w_1 \left(\frac{\frac{1-\pi_0}{\pi_0} e^v}{1 + \frac{1-\pi_0}{\pi_0} e^v} \right) = h_1(v) \qquad \text{if } v \leqslant b. \tag{6}$$

If $v \geqslant a$ we stop and accept H_1 with loss

$$L_\delta(v) = w_0 \left(\frac{1}{1 + \frac{1-\pi_0}{\pi_0} e^v} \right) = h_2(v) \qquad \text{if } a \leqslant v. \tag{7}$$

And, in general, using renewal arguments

$$L_\delta(v) = \begin{cases} h_1(v) & v \leqslant b \\ c(v, k(v)) + E_v\left[L_\delta\left(v + \sum_{j=1}^{k(v)} z_j \right) \right] & b < v < a \\ h_2(v) & a \leqslant v. \end{cases} \tag{8}$$

The notation in (8) requires some explanation. $c(v, k)$ is the cost (nonnegative) of taking k observations at position v, such that $c(v, k)$ increases monotonically beyond bound with increasing k. For our testing problem $c(v, k) = c(k)$, independent of v. However, for the development that follows, we allow this dependence on v. $E_v[\cdot]$ is an expectation over the sum of random variables $\sum_{j=1}^{k(v)} z_j$. The subscript v is present because, aside from the appearance of v in the upper limit of the summation, the distribution of $\sum_{j=1}^{k(v)} z_j$ depends upon π_n which, in turn, depends upon v. In fact if $g_k^i(\cdot)$ is the distribution of $\sum_{j=1}^{k} z_j$ under H_1, then the distribution of $\sum_{j=1}^{k} z_j$ for the purpose of taking the expectation in (8) is

$$g_k^v(\cdot) = \left[\frac{1}{1 + \frac{1-\pi_0}{\pi_0} e^v}\right] g_k^0(\cdot) + \left[\frac{\frac{1-\pi_0}{\pi_0} e^v}{1 + \frac{1-\pi_0}{\pi_0} e^v}\right] g_k^1(\cdot).$$

Just as we have allowed the cost to be generally $c(v, k)$ for this presentation, we also allow h_1 and h_2 to be quite general (integrable) nonnegative functions of v.

If we let $L^*(v)$ be the minimum achievable average loss starting at v over all scheduling rules, we arrive at a similar recursive relation

$$L^*(v) = \begin{cases} h_1(v) & v \leqslant b \\ \min\limits_{k \geqslant 1} \left[c(v, k) + E_v\left[L^*\left(v + \sum_{j=1}^{k} z_j\right)\right]\right] & b < v < a \\ h_2(v) & a \leqslant v. \end{cases} \tag{9}$$

As Ehrenfeld [2] points out, the validity of recursive relations (8) and (9) can be shown using Dynamic Programming arguments involving limits of risks and minimum risks achievable using procedures restricted to no more than n stages (see for example Ferguson [3] and Ross [6]).

We shall present convergent iterative procedures allowing us to find $L^*(v)$ and an associated optimal scheduling rule $k^*(v)$, the value of k (the smallest, if there are more than one) achieving the minimum in (9). In the discrete situation (the symmetric binomial test for example) (8) becomes a set of simultaneous linear equations and the iterations leading to k^* and L^* are not difficult to program on a computer (see the numerical examples at the end of this paper). In the continuous case, however, computation becomes much more difficult. For example trying to find $L_\delta(v)$ for a given $k(v)$ (8), involving no searching for optima, necessitates solving a Fredholm integral equation of the second type, a formidable task even to obtain approximate solutions.

3. THE ITERATIVE PROCEDURE

Suppose we let $L^0(v)$ be the average loss until termination, starting at v, achieved by using a given scheduling rule $k^0(v) \geqslant 1$, $b < v < a$. This is the first step of the iterative procedure, and, for the continuous case, perhaps the most computationally difficult one to implement. Now we let $L^1(v)$ be the average loss, scheduling at the first stage of testing not $k^0(v)$, but $k^1(v)$ observations, the best number of observations to schedule, assuming that from the second stage of testing until termination $k^0(v)$ will be

used. Clearly $L^1(v) \leq L^0(v)$. Proceeding in this manner we let $k^2(v)$ be the best number of observations to schedule the first stage of testing, assuming that $k^1(v)$ observations will be scheduled the second-stage, and from the third stage until termination $k^0(v)$ will be used. $L^2(v)$ is the associated average loss. Thus as we continue to define $k^3, L^3, k^4, L^4, \ldots$ the probability that termination occurs before k^0 is put into use increases, and it would appear that $k^i(v)$ and $L^i(v)$ converge to the optima $k^*(v)$ and $L^*(v)$ given by (9) ($k^*(v)$ is the value of k achieving the minimum in (9)).

In the next section we prove that $L^{i+1}(v) \leq L^i(v)$ for all i and thus, since the $L^i(v)$ are all nonnegative, that they converge to some $L(v)$. We further prove that $L(v)$ is indeed $L^*(v)$, the unique solution to (9). We also show that $k^i(v)$ converges to $k^*(v)$.

We first make the definitions of $k^i(v)$ and $L^i(v)$ more precise. Let $k^0(v)$ be a finite, integer valued, nonnegative function for $b < v < a$. Then

$$L^0(v) = \begin{cases} h_1(v) & v \leq b \\ c(v,k^0(v)) + E_v\left[L^0\left(v + \sum_{j=1}^{k^0(v)} z_j\right)\right] & b < v < a \\ h_2(v) & a \leq v \end{cases} \tag{10}$$

and

$$L^{i+1}(v) = \begin{cases} h_1(v) & v \leq b \\ \min\limits_{k \geq 1}\left[c(v,k) + E_v\left[L^i\left(v + \sum_{j=1}^{k} z_j\right)\right]\right] & b < v < a, \\ h_2(v) & a \leq v \end{cases} \tag{11}$$

where $k^{i+1}(v)$ is defined to be that value of k which achieves the minimum in (11).

4. PROOFS OF CONVERGENCE TO OPTIMA

First observe that $L^i(v) \geq 0$ for all i. Thus to show pointwise convergence of $L^i(v)$ to $L(v)$, i.e., $L(v) = \lim\limits_{i \to \infty} L^i(v)$, it suffices to show that $L^{i+1}(v) \leq L^i(v)$ for all i.

THEOREM 2: $L^{i+1}(v) \leq L^i(v)$ for $i = 0, 1, 2, \ldots$ and thus there exists $L(v) = \lim\limits_{i \to \infty} L^i(v)$.

PROOF: For $v \leq b$, $L^i(v) = h_1(v)$ and for a $\leq v$, $L^i(v) = h_2(v)$ for all i from definitions. Thus for $v \notin (b,a)$, $L^{i+1}(v) = L^i(v)$. For $v \in (b,a)$ we use induction on i. First, from the definition of $L^1(v)$ (11) and of $L^0(v)$ (10), we have for $b < v < a$.

$$\begin{aligned} L^1(v) &= \min_{k \geq 1}\left[c(v,k) + E_v\left[L^0\left(v + \sum_{j=1}^{k} z_j\right)\right]\right] \\ &\leq c(v,k^0(v)) + E_v\left[L^0\left(v + \sum_{j=1}^{k^0(v)} z_j\right)\right] \\ &= L^0(v). \end{aligned}$$

Now assume that $L^i(v) \leq L^{i-1}(v)$. From (11) for $b < v < a$ we have

$$L^{i+1}(v) = \min_{k \geq 1}\left[c(v,k) + E_v\left[L^i\left(v + \sum_{j=1}^{k} z_j\right)\right]\right]$$

$$\leq c(v,k^i(v)) + E_v\left[L^i\left(v+\sum_{j=1}^{k^i(v)} z_j\right)\right].$$

And thus from the inductive assumption,

$$L^{i+1}(v) \leq c(v,k^i(v)) + E_v\left[L^{i-1}\left(v+\sum_{j=1}^{k^i(v)} z_j\right)\right] = L^i(v)$$

And since $L^{i+1}(v) = L^i(v)$ for $v \notin (b,a)$, we conclude $L^{i+1}(v) \leq L^i(v)$.

We now show that $L(v) = L^*(v)$, the minimum expected loss starting from v until termination. We do this by showing that $L(v)$ satisfies (9) and noting that, at least from a physical interpretation, (9) has a unique solution.

THEOREM 3: $L(v)$ satisfies (9) and thus equals $L^*(v)$.

PROOF: Since $L^i(v) = h_1(v)$, for $v \leq b$ and $L^i(v) = h_2(v)$, for $a \leq v$, for all i, then clearly

$$L(v) = \begin{cases} h_1(v) & v \leq b. \\ h_2(v) & a \leq v \end{cases}$$

Also taking the limit as $i \to \infty$ in (11) for $b < v < a$

$$L(v) = \lim_{i\to\infty} \min_{k\geq 1}\left[c(v,k) + E_v\left[L^i\left(v+\sum_{j=1}^{k} z_j\right)\right]\right].$$

Thus it suffices to show that

$$\lim_{i\to\infty} \min_{k\geq 1}\left[c(v,k) + E_v\left[L^i\left(v+\sum_{j=1}^{k} z_j\right)\right]\right] = \min_{k\geq 1}\left[c(v,k) + E_v\left[L\left(v+\sum_{j=1}^{k} z_j\right)\right]\right].$$

Now

$$\min_{k\geq 1}\left[c(v,k) + E_v\left[L\left(v+\sum_{j=1}^{k} z_j\right)\right]\right] = c(v,k_0) + E_v\left[L\left(v+\sum_{j=1}^{k_0} z_j\right)\right];$$

i.e., k_0 is one k which achieves the minimum (k_0 should not be confused with $k^0(v)$). Further, for any i

$$\min_{k\geq 1}\left[c(v,k) + E_v\left[L^i\left(v+\sum_{j=1}^{k} z_j\right)\right]\right] \leq c(v,k_0) + E_v\left[L^i\left(v+\sum_{j=1}^{k_0} z_j\right)\right].$$

Thus

$$\lim_{i\to\infty} \min_{k\geq 1}\left[c(v,k) + E_v\left[L^i\left(v+\sum_{j=1}^{k} z_j\right)\right]\right] \leq c(v,k_0) + \lim_{i\to\infty} E_v\left[L^i\left(v+\sum_{j=1}^{k_0} z_j\right)\right].$$

Since the $L^i(v)$ coverage monotonically to $L(v)$, the Monotone Convergence Theorem (see, for example, Royden [7]) gives

$$\lim_{i\to\infty}\min_{k\geq 1}\left[c(v,k)+E_v\left[L^i\left(v+\sum_{j=1}^{k}z_j\right)\right]\right]\leq c(v,k_0)+E_v\left[L\left(v+\sum_{j=1}^{k_0}z_m\right)\right]$$
$$=\min_{k\geq 1}\left[c(v,k)+E_v\left[L\left(v+\sum_{j=1}^{k}z_j\right)\right]\right]$$

establishing inequality in one direction.

To show inequality in the opposite direction, and thus establish the theorem, observe that since the $L^i(v)$ converge monotonically to $L(v)$,

$$c(v,k)+E_v\left[L^i\left(v+\sum_{j=1}^{k}z_j\right)\right]\geq c(v,k)+E_v\left[L\left(v+\sum_{j=1}^{k}z_j\right)\right]$$

for all $i\geq 0$ and all $k\geq 1$. Therefore,

$$\min_{k\geq 1}\left[c(v,k)+E_v\left[L^i\left(v+\sum_{j=1}^{k}z_j\right)\right]\right]\geq\min_{k\geq 1}\left[c(v,k)+E_v\left[L\left(v+\sum_{j=1}^{k}z_j\right)\right]\right]$$

for all $i\geq 0$. Hence, taking the limit as $i\to\infty$

$$\lim_{i\to\infty}\min_{k\geq 1}\left[c(v,k)+E_v\left[L^i\left(v+\sum_{j=1}^{k}z_j\right)\right]\right]\geq\min_{k\geq 1}\left[c(v,k)+E_v\left[L\left(v+\sum_{j=1}^{k}z_j\right)\right]\right].$$

Finally we now show that $k^i(v)$, as defined in (11) "converges" to an optimal scheduling rule, $k^*(v)$.

THEOREM 4: Let K_v be the set of k which achieves the minimum in

$$L(v)=\min_{k\geq 1}\left[c(v,k)+E_v\left[L\left(v+\sum_{j=1}^{k}z_j\right)\right]\right].$$

Then there exists an N_v such that for all $i>N_v$, $k^i(v)\epsilon K_v$.

PROOF: We know that $L(v)$ exists and is finite through known properties of the SPRT. Thus since $c(v,k)\to\infty$ monotonically as k increases, K_v is bounded from above by, say, B_v. For similar reasons we can also find a $B_v'>B_v+1$, such that for $k\geq B_v'$

$$c(v,k)\geq\min_{\substack{1\leq l\leq B_v+1\\ l\notin K_v}}\left[c(v,l)+E_v\left[L\left(v+\sum_{j=1}^{l}z_j\right)\right]\right].$$

Note that the above minimum is over a non-empty set since l is allowed to range up to B_v+1. It is also greater than $L(v)$ by definition of K_v. Now define

$$\epsilon=\min_{\substack{1\leq k\leq B_v'\\ k\epsilon K_v}}\left[c(v,k)+E_v\left[L\left(v+\sum_{j=1}^{k}z_j\right)\right]\right]-L(v)$$

Note that as before the minimum is over a non-empty set, and $\epsilon > 0$ since $k \notin K_v$ and the set is finite. We can rewrite the above as

$$\min_{\substack{1 \le k \le B'_v \\ k \notin K_v}} \left[c(v,k) + E_v \left[L\left(v + \sum_{j=1}^{k} z_j \right) \right] \right] = L(v) + \epsilon.$$

We now suppose the theorem is false and deduce a contradiction. That is, for any N suppose there exists an $i > N$ such that $k^i(v) \notin K_v$. For each such i

$$L^i(v) = c(v,k^i(v)) + E_v \left[L^{i-1} \left(v + \sum_{j=1}^{k^i(v)} z_j \right) \right]$$

$$\geq c(v,k^i(v)) + E_v \left[L \left(v + \sum_{j=1}^{k^i(v)} z_j \right) \right],$$

since $L^i(v) \geq L(v)$ for all i. But now either $k^i(v) \leq B'_v$ or $k^i(v) > B'_v$. If $k^i(v) \leq B'_v$, then, since by assumption $k^i(v) \notin K_v$, we have

$$L^i(v) \geq c(v,k^i(v)) + E_v \left[L \left(v + \sum_{j=1}^{k^i(v)} z_j \right) \right]$$

$$\geq \min_{\substack{1 \le k \le B'_v \\ k \notin K_v}} \left[c(v,k) + E_v \left[L \left(v + \sum_{j=1}^{k} z_j \right) \right] \right]$$

$$= L(v) + \epsilon.$$

If $k^i(v) > B'_v$ then by definition of B'_v

$$L^i(v) \geq c(v, k^i(v)) + E_v \left[L \left(v + \sum_{j=1}^{k^i(v)} z_j \right) \right]$$

$$\geq c(v, k^i(v))$$

$$\geq \min_{\substack{1 \le k \le B_v + 1 \\ k \notin K_v}} \left[c(v, k) + E_v \left[L \left(v + \sum_{j=1}^{k} z_j \right) \right] \right]$$

$$\geq \min_{\substack{1 \le k \le B'_v \\ k \notin K_v}} \left[c(v, k) + E_v \left[L \left(v + \sum_{j=1}^{k} z_j \right) \right] \right]$$

$$= L(v) + \epsilon$$

since B'_v was chosen $> B_v + 1$. In either case $L^i(v) \geq L(v) + \epsilon$. Thus we have shown that for any

N there exists an $i > N$ such that $L^i(v) \geq L(v) + \epsilon$ with $\epsilon > 0$. This contradicts the convergence of $L^i(v)$ to $L(v)$.

Note that this last theorem is a rather strong result since it shows that for each v, $k^i(v)$ reaches $k^*(v)$ in a finite number of iterations and remains there. For the discrete case in which v can take on only a finite number of values in the interval (b, a), we may conclude the uniform result that the functions $k^i(v)$ will reach and remain at $k^*(v)$ for all $i > M$, where $M = \max_{b<v<a} N_v$. However, the same cannot be said about the associated $L^i(v)$ reaching $L^*(v)$ without the use of value determination, discussed in the next section.

5. VALUE DETERMINATION

Up to now each step of the iterative procedure, other than the initial one of finding L^0 given k^0, has been of the same type—one commonly named "policy improvement." We have shown that this procedure, involving only policy improvement steps, converges to the optimal $L^*(v)$ and $k^*(v)$. However, we can also add a value determination step after each, or any one, policy improvement step and still maintain convergence to the optima. The addition of a value determination step after each policy improvement step makes the procedure resemble very closely those discussed for the discrete case in references cited earlier, [2, 5]. Based upon computational experience in the discrete case the addition of value determination increases the rapidity of convergence. However, in the continuous case each value determination step requires the solution of the same type of integral equation necessary in solving for $L^0(v)$ given $k^0(v)$.

More precisely, at some step, having calculated $L^i(v)$ and $k^i(v)$, instead of proceeding as before with another policy improvement step using (11) to obtain $L^{i+1}(v)$ and $k^{i+1}(v)$, we may carry out a value determination step by solving for $\mathscr{L}^i(v)$ in

$$\mathscr{L}^i(v) = \begin{cases} h_1(v) & v \leq b \\ c(v, k^i(v)) + E_v\left[\mathscr{L}^i\left(v + \sum_{j=1}^{k^i(v)} z_j\right)\right] & b < v < a \\ h_2(v) & a \leq v. \end{cases} \tag{12}$$

If we can show that $\mathscr{L}^i(v) \leq L^i(v)$, then we may consider $\mathscr{L}^i(v)$ and $k^i(v)$ to be new $L^0(v)$ and $k^0(v)$ and proceed with a policy improvement step from there using (11) without affecting convergence.

THEOREM 5: $\mathscr{L}^i(v) \leq L^i(v)$

PROOF: (Based upon results by Ehrenfeld, [2].) From the definition of $k^i(v)$

$$L^{i-1}(v) \geq L^i(v) = c(v, k^i(v)) + E_v\left[L^{i-1}\left(v + \sum_{j=1}^{k^i(v)} z_j\right)\right].$$

Thus,

$$L^i(v) \geq c(v, k^i(v)) + E_v\left[L^i\left(v + \sum_{j=1}^{k^i(v)} z_j\right)\right]. \tag{13}$$

Now define $\Delta(v) = L^i(v) - \mathscr{L}^i(v)$. From the definitions of $L^i(v)$ and $\mathscr{L}^i(v)$

$$\Delta(v) = \begin{cases} 0 & v \leq b \\ L^i(v) - c(v, k^i(v)) - E_v\left[\mathscr{L}^i\left(v + \sum_{j=1}^{k^i(v)} z_j\right)\right] & b < v < a \\ 0 & a \leq v. \end{cases} \tag{14}$$

From inequality (13) we have

$$\theta(v) = L^i(v) - c(v, k^i(v) - E_v\left[L^i\left(v + \sum_{j=1}^{k^i(v)} z_j\right)\right] \geq 0.$$

Now substitution $L^i(v) - \Delta(v)$ for $\mathscr{L}^i(v)$ in (14) we obtain

$$\Delta(v) = \begin{cases} 0 & v \leq b \\ \theta(v) + E_v\left[\Delta\left(v + \sum_{j=1}^{k^i(v)} a_j\right)\right] & b < v < a \\ 0 & a \leq v. \end{cases}$$

This equation allows us to view $\Delta(v)$ as the expected average loss until termination when the cost function is $\theta(v) \geq 0$ and there is zero terminal loss, and the scheduling rule $k^i(v)$ is used. The equation has a unique solution with $\Delta(v) \geq 0$.

In the previous section we commented on the fact that $k^i(v)$ can reach $k^*(v)$ for all values of v without the corresponding $L^i(v)$ reaching $L^*(v)$. This situation is remedied with the use of value determination. Specifically suppose at some step we have $L^i(v)$ and $k^i(v)$ and then carry out a value determination to obtain $\mathscr{L}^i(v)$. If we then carry out a policy improvement on $\mathscr{L}^i(v)$ to obtain $\hat{L}^{i+1}(v)$ and $\hat{k}^{i+1}(v)$, and $\hat{k}^{i+1}(v) = k^i(v)$ for all v, then it is not difficult to show that $\hat{k}^{i+1}(v) = k^i(v) = k^*(v)$ for all v as well as $\hat{L}^{i+1}(v) = \mathscr{L}^i(v) = L^*(v)$ for all v, and all subsequent iterations will remain at the optimum.

6. SIX SUBOPTIMAL TESTING PROCEDURES

In this section we present and compare six suboptimal testing procedures, derived primarily from the iterative methods previously developed. Their derivation involves only one or two iterations, and thus they are easier to find than the optimal procedure. Further, the first iterate is chosen in such a way as to provide a start that is usually not very far from optimal. Specifically, the first iterate involves using the best constant scheduling rule, $k^0(v)$ equals a constant. In the specific case of the symmetric binomial test, in which case the problem becomes discrete, all six suboptimal procedures were found to yield average losses quite close to, and some were actually equal to, the minimum losses. These numerical results appear in the next section.

The six procedures are as follows:

1) As a function of the starting position, v, find the optimum constant scheduling rule, or batch size, $k_0(v)$ and the associated average loss, $L_0(v)$. The testing procedure typically starts at $v = 0$, and thus $k_0(0)$ is used from start to finish, achieving $L_0(0)$. That is,

$$L_0(v) = \min_{k \geq 1} J_k(v), \tag{15}$$

where $k_0(v)$ achieves the minimum and $J_k(v)$ satisfies

$$J_k(v)=\begin{cases} h_1(v) & v \leq b \\ c(v,k)+E_v\left[J_k\left(v+\sum_{j=1}^{k} z_j\right)\right] & b<v<a. \\ h_2(v) & a \leq v \end{cases} \tag{16}$$

2) Obtain $k_0(v)$ as in procedure 1), but when testing revise the best constant batch size to use at each stage. That is, schedule $k_0(v)$ observations at each stage as a function of the position, v. The associated average loss, $L_1(v)$, satisfies

$$L_1(v)=\begin{cases} h_1(v) & v \leq b \\ c(v,k_0(v))+E_v\left[L_1\left(v+\sum_{j=1}^{k_0(v)} z_j\right)\right] & b<v<a \\ h_2(v) & a \leq v. \end{cases} \tag{17}$$

3) As a function of the starting position, v, find the best $k_1(v)$ to schedule the first stage, assuming that from the second stage until termination the best constant batch size will be used as determined by the resulting v at the second stage. Let the associated average loss be $L_2(v)$. The testing procedure starts at $v=0$ with $k_1(0)$ observations scheduled the first stage. From the second stage until termination the constant batch size $k_0(v)$ is used, v being the position at the start of the second stage. That is

$$L_2(v)=\begin{cases} h_1(v) & v \leq b \\ \min\limits_{k\geq 1}\left[c(v,k)+E_v\left[L_0\left(v+\sum_{j=1}^{k} z_j\right)\right]\right] & b<v<a \\ h_2(v) & a \leq v. \end{cases} \tag{18}$$

$k_1(v)$ achieving the minimum.

4) Obtain $k_1(v)$ as in procedure 3), but use it throughout the testing procedure. That is, at each stage schedule $k_1(v)$ observations as a function of the position, v. The associated average loss, $L_3(v)$, satisfies

$$L_3(v)=\begin{cases} h_1(v) & v \leq b \\ c(v,k_1(v))+E_v\left[L_3\left(v+\sum_{j=1}^{k_1(v)} z_j\right)\right] & b<v<a \\ h_2(v) & a \leq v. \end{cases} \tag{19}$$

5) Procedure 5) is similar to procedure 3) with $L_1(v)$ replacing $L_0(v)$. That is, we find $k_2(v)$, the best number of observations to schedule the first stage, assuming that from the second stage until termination the scheduling rule $k_0(v)$ will be used. The associated average loss, $L_4(v)$, is given by

$$L_4(v)=\begin{cases} h_1(v) & v \leq b \\ \min\limits_{k\geq 1}\left[c(v,k)+E_v\left[L_1\left(v+\sum_{j=1}^{k} z_j\right)\right]\right] & b<v<a \\ h_2(v) & a \leq v. \end{cases} \tag{20}$$

$k_2(v)$ achieving the minimum.

6) Procedure 6) is, analogously, similar to procedure 4). $k_2(v)$ is obtained as in procedure 5) but is adhered to throughout the testing. The associated average loss, $L_5(v)$, satisfies

$$L_5(v) = \begin{cases} h_1(v) & v \leq b \\ c(v, k_2(v)) + E_v\left[L_5\left(v + \sum_{j=1}^{k_2(v)} z_j\right)\right] & b < v < a \\ h_2(v) & a \leq v. \end{cases} \tag{21}$$

Observe that from Theorems 2 and 5 we know that $L_5(v) \leq L_4(v) \leq L_1(v) \leq L_0(v)$ and that $L_3(v) \leq L_2(v)$. We now prove further that $L_2(v) \leq L_0(v)$ and also that $L_4(v) \leq L_2(v)$. Clearly from definitions all six average losses are equal for $v \notin (b, a)$ so that we need only consider $v \in (b, a)$.

THEOREM 6: $L_2(v) \leq L_0(v)$.

PROOF: Let k_c be the constant batch size used in procedure 1) to achieve $L_0(v)$ ($k_c = k_0(v)$), fixed throughout the testing procedure that is $L_0(v) = J_{k_c}(v)$ for $b < v < a$. Now from (15) $L_0(v) \leq J_k(v)$ for any v, and $k \geq 1$. Thus from (18), for $b < v < a$ we have

$$\begin{aligned} L_2(v) &\leq c(v, k_c) + E_v\left[L_0\left(v + \sum_{j=1}^{k_c} z_j\right)\right] \\ &\leq c(v, k_c) + E_v\left[J_{k_c}\left(v + \sum_{j=1}^{k_c} z_j\right)\right] \\ &= L_0(v). \end{aligned}$$

THEOREM 7: $L_4(v) \leq L_2(v)$.

PROOF: Since we know $L_1(v) \leq L_0(v)$ for all v, we have

$$c(v, k) + E_v\left[L_1\left(v + \sum_{j=1}^{k} z_j\right)\right] \leq c(v, k) + E_v\left[L_0\left(v + \sum_{j=1}^{k} z_j\right)\right]$$

for all $k \geq 1$. Thus it follows that

$$\min_{k \geq 1}\left[c(v, k) + E_v\left[L_1\left(v + \sum_{j=1}^{k} z_j\right)\right]\right] \leq \min_{k \geq 1}\left[c(v, k) + E_v\left[L_0\left(v + \sum_{j=1}^{k} z_j\right)\right]\right].$$

The theorem follows from definitions (18) and (20).

7. NUMERICAL EXAMPLES, THE SYMMETRIC BINOMIAL CASE

Suppose that each X_i can take on only two values, 0 or 1, with $Pr(X_i = 0) = 1 - p$ and $Pr(X_i = 1) = p$. We wish to test $H_0: p = p_0$ vs. $H_1: p = p_1 = 1 - p_0 > p_0$. Under these assumptions

$$P_0(x) = p_0^x (1 - p_0)^{1-x}$$

$$P_1(x) = (1 - p_0)^x p_0^{1-x}$$

so that

$$z = \ln \frac{P_1(x)}{P_0(x)} = W(2x - 1),$$

where $W = \ln \frac{1-p_0}{P_0} > 0$. Thus

$$z = \begin{cases} -W & \text{with probability, } Pr(X=0) \\ +W & \text{with probability, } Pr(X=1) \end{cases}$$

so that the log-likelihood ratio, $v = \sum_{j=1}^{n} z_j$, can take on, within the interval (b, a), only a finite number of values $-n_bW$, $(-n_b+1)W$, . . ., $-W$, O, W, . . ., $(n_a-1)W$, n_aW, where n_b is the greatest integer less than or equal to $\frac{|b|}{W}$ and n_a is the greatest integer less than or equal to $\frac{a}{W}$. Therefore we need only be concerned with $L(v)$ and $k(v)$ for these n_a+n_b+1 values of v. If we set $h_1(v) = h_2(v) = 0$, Equations (8) then reduce to a set of n_a+n_b+1 linear equations in the n_a+n_b+1 unknowns $L(-n_bW)$, . . ., $L(n_aW)$, the coefficients being convex combinations of binomial probabilities. We set $h_1(v) = h_2(v) = 0$ for computational facility, and thus $L(v)$ is now the expected sampling costs, assuming losses due to incorrect terminal decisions are taken care of by judicious choice of b and a.

With the aid of a digital computer we computed the losses and scheduling rules for the six procedures of the last section, as well as the optimal values, for several cases. Three cases are presented in Tables 1, 2, and 3. In all three cases c was chosen to be 1 without loss of generality, and π was chosen to be ½. An experimentor, starting from scratch, wishing to know his expected losses until termination, would look under the column $v = 0$. The tables offer some flexibility in that if one wanted to begin with π not equal to ½ but instead $\frac{1}{1+e^{nW}}$, and b and a were replaced by $b+nW$ and $a+nW$ respectively for $n = -n_b$, . . ., n_a, then to find one's expected losses from start until termination one would look under the column $v = nW$.

The primary purpose of these numerical examples is to illustrate the six suboptimal procedures discussed in the previous section. Thus for all three of the cases the first iteration step was the con-

TABLE 1

$b = -5.0$ $a = 5.0$ $p_0 = 0.3$ $d = 9.0$ $c = 1.0$

Therefore $W = 0.847$, $n_b = n_a = 5$, a total of 11 points.

$v = \pm 5W$	$\pm 4W$	$\pm 3W$	$\pm 2W$	$\pm 1W$	0	
5	6	9	12	15	14	$k_0(v)$
19.13	22.53	27.64	30.93	34.75	35.20	$L_0(v)$
18.12	21.93	26.33	30.24	33.01	34.07	$L_1(v)$
1	6	9	12	15	16	$k_1(v)$
16.89	22.12	26.55	30.48	33.25	34.25	$L_2(v)$
16.70	21.93	26.32	30.23	33.01	34.01	$L_3(v)$
1	6	9	12	15	16	$k_2(v)$
16.71	21.93	26.33	30.24	33.01	34.01	$L_4(v)$
16.70	21.93	26.32	30.23	33.01	34.01	$L_5(v)$

NOTE: $k_1(v)$ and $k_2(v)$ are both the optimum scheduling rules and thus $L_3(v)$ and $L_5(v)$ are the minimum expected losses.

stant batch size procedure to obtain $L_0(v)$. The advantage of such a first step over an arbitrary initial scheduling rule accounts for the rapidity of convergence and hence the relatively small amounts of improvement in going from $L_0(v)$ to $L^*(v)$, although in Table 3 an improvement of 10–15 percent can be observed. More dramatic improvements with each iteration are observed, in general, when arbitrary initial scheduling rules are used. The tables demonstrate the relationships among the six suboptimal procedures as derived in the previous section. As can be seen all six procedures yield good, if not optimal, results, even the constant batch size procedure, $L_0(v)$. The expected loss, were observations scheduled one at a time as in the SPRT, for $v=0$, is 148.15 for Table 1, 138.13 for Table 2, and 321.68 for Table 3. The improvement over these values of any of the six procedures is obvious.

TABLE 2

$b=-2.5$ $a=7.5$ $p_0=0.3$ $d=9.0$ $c=1.0$

Therefore $W=0.847$, $n_b=2$, $n_a=8$, a total of 11 points.

$v=-2W$	$-1W$	0	$1W$	$2W$	$3W$	$4W$	$5W$	$6W$	$7W$	$8W$	
7	10	15	16	17	14	13	10	9	6	5	$k_0(v)$
25.77	31.78	37.84	38.59	38.34	35.20	32.75	28.56	25.74	21.35	18.49	$L_0(v)$
24.03	31.07	35.88	37.11	36.11	34.03	31.13	27.98	24.67	20.98	17.67	$L_1(v)$
1	6	15	18	17	16	13	10	7	6	1	$k_1(v)$
21.51	30.72	36.20	37.35	36.36	34.19	31.35	28.18	24.79	21.10	16.41	$L_2(v)$
20.87	30.03	35.81	37.05	36.08	33.95	31.11	27.97	24.61	20.98	16.30	$L_3(v)$
1	6	15	18	17	16	13	10	7	6	1	$k_2(v)$
21.25	30.24	35.88	37.09	36.11	33.97	31.13	27.98	24.61	20.98	16.30	$L_4(v)$
20.87	30.03	35.81	37.05	36.08	33.95	31.11	27.97	24.61	20.98	16.30	$L_5(v)$

NOTE: $k_1(v)$ and $k_2(v)$ are both the optimum scheduling rule and thus $L_3(v)$ and $L_5(v)$ are the minimum expected losses.

TABLE 3

$b=-5.0$ $a=5.0$ $p_0=0.4$ $d=4.0$ $c=1.0$

Therefore $W=0.405$, $n_b=n_a=12$, a total of 25 points.

$v=\pm12W$	$\pm11W$	$\pm10W$	$\pm9W$	$\pm8W$	$\pm7W$	$\pm6W$	$\pm5W$	$\pm4W$	$\pm3W$	$\pm2W$	$\pm1W$	0	
3	6	9	10	13	14	17	18	19	20	21	20	21	$k_0(v)$
20.25	25.81	35.02	40.25	49.88	55.10	65.13	70.15	80.04	83.80	91.99	92.55	96.77	$L_0(v)$
16.51	24.34	31.40	37.93	44.77	51.63	58.51	65.48	72.26	78.43	83.44	87.12	88.12	$L_1(v)$
1	4	7	10	13	16	21	24	29	34	37	40	41	$k_1(v)$
15.36	24.67	32.26	39.47	46.57	53.68	60.79	67.78	74.42	80.28	84.85	87.76	88.76	$L_2(v)$
14.32	23.21	30.49	37.39	44.16	50.93	57.85	64.49	70.98	76.76	81.20	84.12	85.12	$L_3(v)$
1	4	7	10	13	16	19	22	25	30	33	36	37	$k_2(v)$
14.77	23.70	31.01	37.93	44.77	51.62	58.48	65.26	71.74	77.49	82.05	84.95	85.95	$L_4(v)$
14.32	23.20	30.47	37.35	44.12	50.87	57.62	64.29	70.66	76.38	80.89	83.80	84.80	$L_5(v)$
1	2	5	8	11	14	17	22	25	28	33	36	37	$k^*(v)$
14.31	23.18	30.43	37.30	44.06	50.82	57.58	64.26	70.63	76.33	80.86	83.77	84.77	$L^*(v)$

NOTE: The optimum $k^*(v)$ and $L^*(v)$ were obtained from one more iteration and value determination from $L_5(v)$.

REFERENCES

[1] Derman, C., "On Sequential Decisions and Markov Chains," Management Science *9*, 16–23 (1967).

[2] Ehrenfeld, S., "On Group Sequential Sampling," Technometrics, Vol. 14, No. 1 (1972).

[3] Ferguson, T. S., *Mathematical Statistics, A Decision Theoretic Approach* (Academic Press, Inc., New York, 1967).

[4] Lehmann, E. L., *Testing Statistical Hypotheses* (John Wiley and Sons, Inc., New York, 1959).

[5] Osaki, S. and Mine, H., "Some Remarks on a Markovian Decision Problem with an Absorbing State," J. Math. Anal. and Appl. *23*, 327–333 (1968).

[6] Ross, S. M., *Applied Probability Models with Optimization Applications* (Holden-Day, San Francisco, 1970).

[7] Royden, H. L., *Real Analysis* (The MacMillan Company, New York, 1963).

[8] Wald, A., *Sequential Analysis* (John Wiley and Sons, Inc., New York, 1947).

[9] Wald, A. and Wolfowitz, J., "Optimum Character of the Sequential Probability Ratio Test," Ann. Math. Statist. *19*, 326–339 (1948).

[10] Wald, A. and Wolfowitz, J., "Bayes Solutions of Sequential Decision Problems," Ann. Math. Statist. *21*, 89–99 (1950).

[11] Wetherill, G. B., *Sequential Methods in Statistics* (Methuen and Co., Ltd., London, 1966).

A PROPORTIONAL DEFENSE MODEL

K. C. Shumate

Marine Corps Development and Education Command
Quantico, Virginia

and

G. T. Howard

Naval Postgraduate School
Monterey, California

ABSTRACT

This paper considers the problem of defending a set of point targets of differing values. The defense is proportional in that it forces the offense to pay a price, in terms of reentry vehicles expended, that is proportional to the value of the target. The objective of the defense is to balance its resources so that no matter what attack is launched, the offense will have to pay a price greater than or equal to some fixed value for every unit of damage inflicted. The analysis determines which targets should be defended and determines the optimal firing doctrine for interceptors at defended targets. A numerical example is included showing the relationship between the total target damage and the size of the interceptor force for different values of p, the interceptor single shot kill probability. Some generalizations are discussed.

I. INTRODUCTION

We consider the local defense of a set of point targets of varied value in which the role of the defense is to minimize total damage during a sequential attack by an offensive force of nuclear ballistic missiles. We assume a constant single shot kill probability $p < 1$ associated with one interceptor against one reentry vehicle. The offense receives no information concerning the destruction of any target, the defense does not know the planned size of the attack at any target or the total attack size, and each target can be defended only by the interceptors placed immediately around it. We seek the minimax allocation of interceptor missiles based on the assumption that the offense knows both the allocation of interceptors and the interceptor commitment policy or firing doctrine at each target. This is the offense-last-move allocation for a group of targets with different values.

The literature on missile defense problems has generally treated the reentry vehicle-interceptor engagement to be one-on-one. The reasons for this are discussed by Eckler and Burr [2] in chapters 4 and 5 of their recent survey of the missile allocation literature. However, for $p < 1$, and for the criterion of defense used in this paper, it is necessary to relax the one-on-one assumption for certain high valued targets and to examine explicitly the defensive interceptor commitment policy or firing doctrine. The model used requires that the maximum average damage per attacker not exceed some fixed level. Battle [1] considered defense of area targets with imperfect interceptors using as the criterion the minimization of the maximum average damage per attacker, but he considered only fixed salvos of

interceptors rather than a variable interceptor commitment policy. Everett [3] also considers the damage per reentry vehicle and examines the interceptor commitment policy for a single target when $p < 1$. He shows that for a given allocation of interceptors at a target, the minimax approach allows an offensive payoff that is typically no more than 10 percent greater than the mixed strategy payoff.

In ballistic missile defense studies, a useful concept is that of the price of a target. We define price to be r/K where r is the number of reentry vehicles sent against a target and K is the probability that the target is killed by the r reentry vehicles. Thus K is a function of r. We can speak of the price that the offense "pays" to "buy" the target. The offensive payoff λ at a single target is the value V of the target times the reciprocal of price: $\lambda = KV/r$.

A proportional defense is one in which the defense forces the offense to pay a price that is proportional to the value of a target. For certain sets of targets, particularly those in which many defended targets are not attacked by the optimal offense, proportional defense will minimize target damage for a fixed force of interceptors. We will assume that proportional defense is optimal in this sense for the set of targets under consideration. Our method of requiring price to be proportional to value will be to allocate interceptors in such a manner that the offensive payoff is the same at each defended target. Rather than allocate a fixed force of interceptors, we will minimize the number of interceptors required to force the offensive payoff to be less than or equal to a fixed payoff λ^*.

II. THE MODEL

For each target the defense, unaware of the planned size of the attack, must determine the number of interceptors I to place around that target and the number of interceptors to be fired against each reentry vehicle. We denote the interceptor firing doctrine by $D = (i_1, \ldots, i_m, \ldots, i_M)$, where i_m is the number of interceptors to be fired against the mth reentry vehicle, and M is the number of reentry vehicles against which the defense will fire. It will become clear that M depends only on λ^* and the value of the target being defended. We assume for convenience in presentation that the defense never fires more than three interceptors at a reentry vehicle. The model, however, does not require this assumption.

We let p be the interceptor single shot kill probability so that $q = 1 - p$ is the probability that a reentry vehicle survives an encounter with a single reentry vehicle. Thus the probability that a target defended with a firing doctrine D survives an attack of size r is

$$K(r) = 1 - \prod_{m=1}^{r} (1 - q^{i_m}). \tag{1}$$

The assumption is made that reentry vehicles are perfect and that the first one not stopped by the defense kills the entire value V of its target. The (expected) offensive payoff at a single target is given by

$$\lambda = KV/r. \tag{2}$$

We assume that the offense, knowing D and V, wishes to select r to maximize λ.

The method used to arrive at a proportional defense is to select D to minimize the number of interceptors I allocated to a target, subject to $\lambda \leq \lambda^*$. Thus the problem faced by the defense at each

target is

$$\text{(3)} \qquad \begin{cases} \underset{D}{\text{minimize}}\ I = \sum_{i=1}^{M} i_m \\ \text{subject to } KV/r \leq \lambda^*, \qquad r = 1, 2, \ldots . \end{cases}$$

Notice that for a fixed firing doctrine K depends only on r and is thus controlled by the offense. Since the offense knows the defensive firing doctrine, the defense must determine the D that minimizes I and satisfies

$$\text{(4)} \qquad \min_{D} \left\{ \max_{r} KV/r \right\} \leq \lambda^*.$$

III. ANALYSIS.

We will introduce some basic concepts, investigate the criterion for defending a target, and consider targets of varied value to determine the required number of interceptors and the firing doctrine for each.

A. Basic Concepts.

First consider a target of value V. Suppose that I, the number of interceptors at the target, is arbitrarily large but that the firing doctrine calls for firing only one interceptor at each reentry vehicle. The firing doctrine may not be optimal, but is discussed here for illustration. Define the *payoff curve*, $L(r)$ to be $L(r) = K(r) \cdot V$. Notice that since D and p are fixed, K and hence L depend only on r. See Figure 1. The offensive payoff for an attack of size r is the slope of the ray, called the λ-*line*, from the origin to the curve $L(r)$. The slope of the λ-line is KV/r and thus corresponding to each r there is a λ-line, designated $\lambda(r)$. Note that $\lambda(r)$ is not the expected value obtained by the rth reentry vehicle but is the average expected value per reentry vehicle for an attack of size r: $\lambda(r) = KV/r$. Now note that $K(r) = 1 - \prod_{m=1}^{r} (1 - q^{i_m})$ is strictly concave for i_m constant and thus $L(r) = K(r)V$ is strictly concave so $\lambda(r) > \lambda(r+1)$. That is, the offense receives decreasing marginal and average returns for

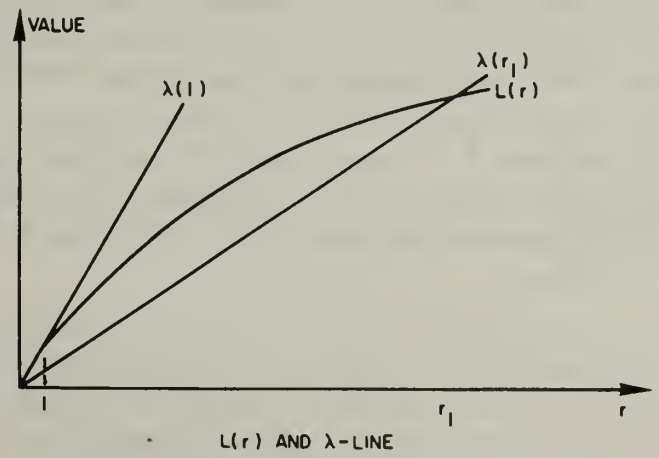


FIGURE 1

increasing r when i_m is constant. Thus to maximize KV/r the optimal attack r^0 for the target and firing doctrine described is to fire one reentry vehicle. Thus $r^0=1$ and $\lambda(r^0)=KV/r$ where $K=1-p$. Then $\lambda=\lambda(r^0)=(1-p)V$ is the payoff at this target.

Define the λ^*-*line* to be a ray from the origin with slope λ. Now recall that the defense's constraint is to force $\lambda \leq \lambda^*$, i.e., $KV/r \leq \lambda^*$. Graphically, the defense must force the slope of the λ-line to be less than λ^* for any size attack. Alternatively, the defense will be satisfied if $L(r)$ is always below the λ^*-line. That is, $L(r) \leq \lambda^* r$. In Figure 2, $L_2(r)$ is a feasible payoff curve since $L_2(r) \leq \lambda^* r$ and $L_1(r)$ is not feasible since there are values of r such that the offensive payoff is greater than λ^*.

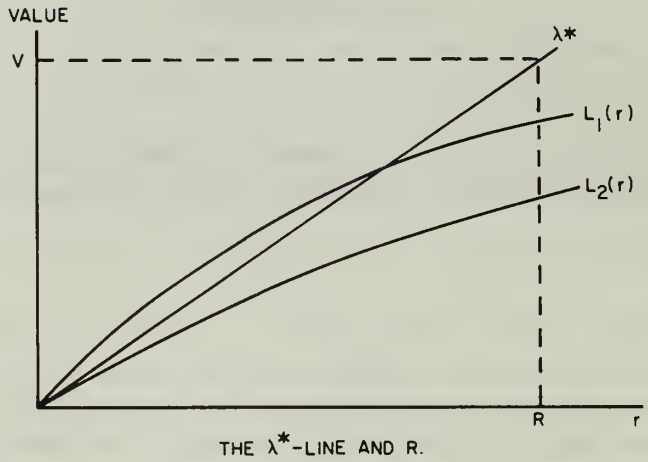


FIGURE 2

B. Criterion for Defending a Target.

Note that if a target has value $V \leq \lambda^*$, $r \geq 1$ yields $\lambda \leq \lambda^*$ and thus the target will not be defended. A target of value greater than λ^* must be defended since if undefended, $r=1$ yields a payoff greater than λ^*.

Consider the situation at any defended target. Define $R=V/\lambda^*$ where V is the value of the target. See Figure 2. Since we will not allow fractional interceptors or reentry vehicles, R must be integer valued, but V/λ^* is not generally integer. Integer considerations are ignored during the analysis for ease of exposition, but numerical calculations use $R=[(V/\lambda^*)+0.999]$ where $[x]$ is the largest integer in x. For an attack of size $r \geq R$, the associated value of the λ^*-line is $\lambda^* r \geq V$. Since the maximum value of $L(r)$ is V, it is clear that for $r \geq R$ the constraint is always satisfied, i.e., $L(r) \leq \lambda^* r$. Thus the defense will never fire at more than $R-1$ reentry vehicles. If the defense were to fire at fewer than $R-1$ reentry vehicles, the $R-1^{st}$ reentry vehicle (or an earlier one, since the offense does not have shoot-look-shoot capability) would destroy the target. Then $\lambda=KV/r$, where $K=1$ and $r=R-1$ ($R>1$ since $V>\lambda^*$ and $R=V/\lambda^*$). Thus

$$\lambda=\frac{V}{(R-1)}=\frac{V}{((V/\lambda^*)-1)}=\lambda^*\frac{V}{V-\lambda^*}>\lambda^*. \tag{5}$$

Thus, the defense must fire against at least $R-1$ reentry vehicles and thus, must always fire at exactly

$R-1$ reentry vehicles. Hence, $M=R-1=\frac{V}{\lambda^*}-1$. We will call an attack of size $r=R$, *exhausting*. For an exhausting attack, the offense will receive a payoff of λ^*. Since the defense will choose a firing doctrine such that the offense never receives more than λ^*, an optimal offensive strategy must be to exhaust the interceptor supply; $r^0=R$.

We have noted that the offense receives decreasing marginal returns for constant i_m, found a criterion for defending a target, determined how many reentry vehicles at which the defense will fire, and found an optimal offense strategy. At any defended target, the only remaining problem is to determine a firing doctrine such that $\lambda \leqq \lambda^*$.

C. Analysis of Firing Doctrine.

We consider now three cases corresponding to increasingly valuable targets. The object is to determine the structure of the optimal firing doctrine.

(1) Small Value Targets: $(1-p)\,V \leqq \lambda^* < V$.

In this case $i_m=1$, $m=1, \ldots, R-1$ is optimal. Note that $\lambda(1)=(1-p)V \leqq \lambda^*$ is feasible, and since the offense receives decreasing returns for $r \leqq R-1$, we have $\lambda(r) \leqq \lambda^* r$, $r=1, \ldots, R-1$. This doctrine is certainly least cost since we must fire at $R-1$ reentry vehicles. The optimal strategy for the offense is to select $r^0=R$. If $(1-p)V=\lambda^*$ an alternate optimum is $r^0=1$. In either case $\lambda(r^0)=\lambda^*$.

(2) Medium Value Targets: $(1-p)^2V \leqq \lambda^* < (1-p)V$.

The value of such a target is too high relative to λ^* for the defense to fire only one interceptor at each reentry vehicle. Notice, for example, if $i_1=1$ the offense can select $r=1$ and receive an expected payoff of $\lambda(1)=(1-p)V > \lambda^*$ which is not permitted in the model. It is feasible to fire two interceptors at each reentry vehicle since, in this case, $\lambda(1)=(1-p)^2V \leqq \lambda^*$. The optimal firing doctrine is obviously to fire two interceptors at as few reentry vehicles as possible then fire one at each reentry vehicle up to and including the $R-1$st.

(3) Large Value Targets: $(1-p)^3V \leqq \lambda^* < (1-p)^2V$.

Reasoning as in (b), this target is too valuable relative to λ^* to fire only two interceptors at each reentry vehicle, and the defense must use a firing doctrine of the form $i_m=3$, $m=1, 2, \ldots, B$, $i_m=2$, $m=B+1, \ldots, A$, $i_m=1$, $m=A+1, \ldots, R-1$, where $B>0$.

We show next that the optimal firing doctrine selects the minimum feasible B and then, given B, selects A to be as small as possible. We will denote these minimum quantities by $\hat{B}$ and $\hat{A}$, the firing doctrine by $F(\hat{B}, \hat{A})$, and the associated payoff curve by $\hat{L}(r)$ (see Figure 3). We have

$$\hat{L}(r)=V\cdot\begin{cases}1-(1-q^3)^r & r \leqq \hat{B}\\ 1-(1-q^3)^{\hat{B}}(1-q^2)^{r-\hat{B}} & \hat{B} \leqq r \leqq \hat{A}\\ 1-(1-q^3)^{\hat{B}}(1-q^2)^{\hat{A}-\hat{B}}(1-q)^{r-\hat{A}}, & \hat{A} \leqq r \leqq R-1.\end{cases} \tag{6}$$

It is not feasible to decrease B below $\hat{B}$ so if any firing doctrine is better than the one using $\hat{B}$ and $\hat{A}$ it will have to have $B>\hat{B}$ and $A<\hat{A}$. We will consider the firing doctrine $F(B_1, A_1)$ where $B_1=\hat{B}+1$ and $A_1=\hat{A}-2$. This firing doctrine uses fewer interceptors, but we will show it is not feasible. We will do this by showing that if $F(B_1, A_1)$ is feasible, then $F(B_2, A_2)$ where $B_2=B_1-1=\hat{B}$ and $A_2=A_1+1=\hat{A}-1$ is feasible. It is obvious that $F(B_2, A_2)$ is not feasible by the definition of $\hat{A}$. First, $F(B_2, A_2)$

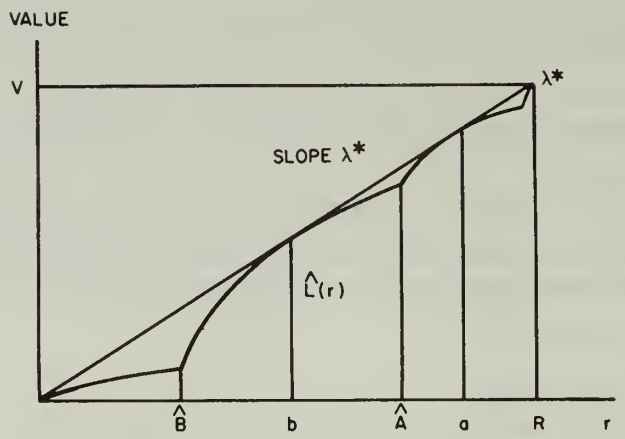


FIGURE 3

is feasible in the range $r \leq \hat{A}-1$ since $L_2(r)$ associated with $F(B_2, A_2)$ is identical to $\hat{L}(r)$. For $\hat{A}-1=A_2 < r \leq R-1$,

$$L_2(r)=V(1-(1-q^3)^{B_2}(1-q^2)^{A_2-B_2}(1-q)^{r-A_2}) \tag{7}$$

and

$$L_1(r)=V(1-(1-q^3)^{B_1}(1-q^2)^{A_1-B_1}(1-q)^{r-A_1}). \tag{8}$$

We now show $L_2(r) < L_1(r)$ for $A_2 < r \leq R-1$ which proves that $F(B_2, A_2)$ is feasible if $F(B_1, A_1)$ is. We have

$$L_1(r)-L_2(r)=V[(1-q^3)^{B_2}(1-q^2)^{A_1-B_1}(1-q)^{r-A_2}((1-q^2)^2-(1-q^3)(1-q))]>0 \tag{9}$$

since

$$(1-q^2)^2-(1-q^3)(1-q)>0.$$

Thus $F(B_1, A_1)$ is not feasible. A similar argument applies to $B_3=\hat{B}+j$, $A_3=\hat{A}-(j+1)$, $j+1=2, \ldots, \hat{A}$.

Thus the defense fires three interceptors at the first $\hat{B}$ reentry vehicles, where $\hat{B}$ is as small as possible; then fires two interceptors at the next $\hat{A}-\hat{B}$ reentry vehicles, where $\hat{A}$ is as small as possible; and then fires one interceptor up to the $R-1^{st}$ reentry vehicle at which point the interceptor supply is exhausted. It is clear again that the optimal offense strategy is $r^0=R$, with alternate optima at $r^0=a$, $r^0=b$, and if $(1-p)^3V=\lambda^*$, at $r^0=1$. If one wished to consider more than three interceptors per reentry vehicle the analysis would be similar. In general, the maximum number of interceptors per reentry vehicle that needs to be considered is the largest integer j such that $\lambda^* \geq V(1-p)^j$ for all targets.

IV. AN EXAMPLE.

For any level of allowable offensive payoff, targets of value close to λ^* will be allocated only a few interceptors. Considering costs of land, radar, etc., it is not an efficient policy to set up a defensive complex for the purpose of defending against a small number of reentry vehicles. One way to "solve" this problem is to agree arbitrarily to defend a certain number of targets, or equivalently to defend targets above a certain value. We have chosen the latter course and the defense is of course no longer exactly proportional.

We consider the defense of 100 targets of various values. The target list used in the example approximates the target structure of the United States, with all cities over 150,000 being defended and large cities broken into several targets. No consideration is taken of overlapping interceptor coverage or of target vulnerability, population values are approximate, and population is the only measure of value. Parametric studies of λ^* and p were conducted to determine changes in the firing doctrine and allocation of interceptors to targets, and to investigate total offense and defense requirements.

Table I shows the firing doctrine for 10 selected combinations of target value λ^*, and single shot kill probability p. Table II shows the total defensive interceptor requirement for various values of λ^* and p. Table II also shows, for various values of λ^*, the total number of reentry vehicles ΣR required by the offense to attack all 100 targets with exhausting attacks of size $r=R$.

Since R is now required to be an integer, $\lambda(R)$ is typically *less* than λ^*, and it is no longer the case that the optimal offensive strategy is to use $r=R$. For example, at target K (see Table I), $R=15$ but the optimal attack has been computed to be $r^0=10$. The payoff for the exhausting attack is $\lambda(R)=$

TABLE I. *Allocation and Firing Doctrine for Selected Targets*

Target	Value·10^{-3}	p	$\lambda^*\cdot10^{-3}$	Fire 3 at first	Fire 2 at next	Fire 1 at next	I	R
A	2647	0.80	22	71	40	9	302	121
B	2647	0.90	22	3	93	24	219	121
C	1424	0.80	48	1	20	8	51	30
D	800	0.90	22	0	16	20	52	37
E	647	0.80	22	1	20	8	51	30
F	647	0.90	22	0	11	18	40	30
G	450	0.92	24	0	2	16	20	19
H	422	0.90	24	0	3	14	20	18
I	422	0.92	24	0	2	15	19	18
J	410	0.90	24	0	3	14	20	18
K	380	0.80	26	0	6	8	20	15

TABLE II. *Defensive and Offensive Requirements for 100-Target Problem*

$\lambda^*\cdot10^{-3}$	ΣR	$p=$ 0.8	0.9	0.92	0.94	0.96
22	2211	3778	2895	2735	2554	2364
26	1879	3040	2355	2229	2085	1943
30	1630	2537	1969	1862	1754	1644
40	1231	1727	1373	1310	1244	1178
50	997	1296	1048	1001	957	918
		Total defensive interceptor requirement				

25,330 and the payoff for the optimal attack is $\lambda(r^0) = 25{,}960$. Notice, however, that the payoff with $r = r^0$ is less than λ^* as required and is only slightly greater than $\lambda(R)$. Furthermore, if the defense desired that only exhausting attacks be optimal (so that the offense must attack fewer targets), only small increases in A and B will be required to force the offense to set $r = R$ at all targets.

V. GENERALIZATION.

A. Discussion.

We previously dealt only with point targets and assumed that reentry vehicles were perfect. In a more general model we can remove these restrictions and deal as well with a number of other considerations, such as mutual interference between interceptors, radar clutter, target hardness, and others. In each of the generalizations discussed, independence of the payoff curves for various targets is assumed.

Here we assume that for each target we have expected damage functions such as those shown in Figure 4, where $L_i(r)$ is the expected target damage caused by an attack of size r given that each reentry vehicle is attacked by i interceptors. The undefended damage function is given by $L_0(r)$. The damage functions $L_i(r)$ can conveniently be interpreted as representing an area target where the expected damage depends on r and the firing doctrine.

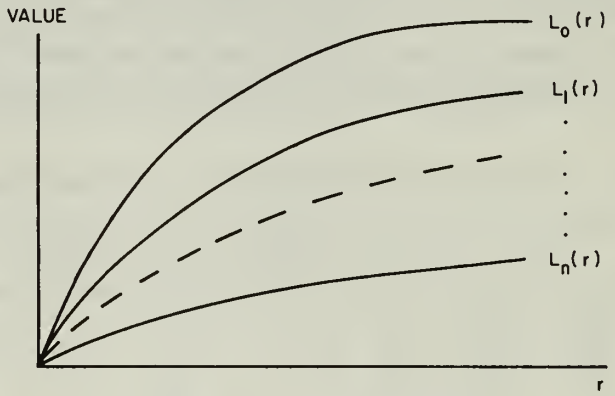


FIGURE 4. Target damage functions

Notice that we no longer assume that the first penetrating reentry vehicle destroys the target. In that case $L_0(r)$ would be a step function, jumping from 0 to V at $r = 1$. Previously, we assumed that $i \leqslant 3$. Here we make no such assumption and denote the largest value of i by n.

The actual damage response function $L(r)$ for the target depends upon the firing doctrine used, and we assume that $L(r)$ is the envelope obtained by shifting the appropriate $L_i(r)$ functions to the right as shown by the solid line in Figure 5. The firing doctrine used in Figure 5 is given by

$$\begin{aligned} i_m &= 2, && m = 1, 2, \ldots, n_2 \\ i_m &= 1, && m = n_2 + 1, \ldots, n_1 \\ i_m &= 0, && m > n_1. \end{aligned}$$

In what follows we make no assumption about the form of the functions $L_i(r)$ at each target, except that $L_i(r) \geqslant L_{i+1}(r)$, and in this case it is not necessarily true that the optimal firing doctrine has $i_1 \geqslant i_2 \geqslant \ldots \geqslant i_M$. We discuss next how dynamic programming can be used the optimal firing doctrine at each target.

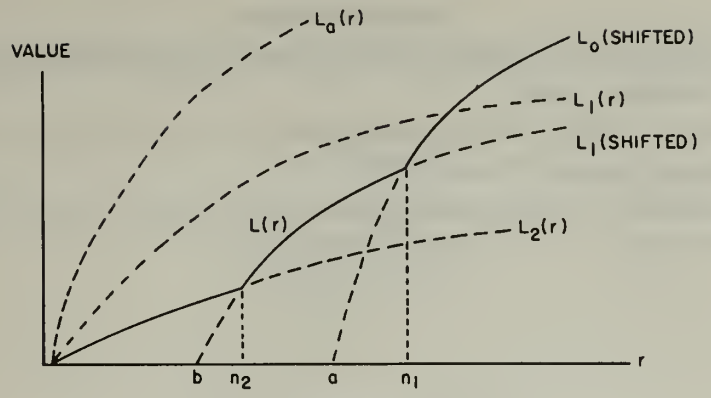


FIGURE 5. Damage response function

B. Dynamic Programming Formulation.

At any target we imagine the defensive decision problem to consist of $R-1$ stages, the decision being the number of interceptors to fire at each reentry vehicle. We let

x_m = the expected level of damage already sustained just before the arrival of the mth reentry vehicle,

i_m = the number of interceptors to fire at the mth reentry vehicle,

$f_m(x_m)$ = min (feasible) number of interceptors required to defend the target against the reentry vehicles $m, \ldots, R$ given that the target damage already sustained is x_m.

Note that $f_R(x_R)=0$ since it is unnecessary to defend against the Rth reentry vehicle. The recursive equations for the dynamic programming formulation are given below.

$$f_m(x_m)=\min_{i_m}\{i_m+f_{m+1}(x_{m+1})\}, \qquad m=1, \ldots, R-1 \tag{10}$$
$$i_m=0, 1, 2, \ldots$$

subject to

$$x_{m+1}=L_{i_m}(L_{i_m}^{-1}(x_m)+1) \leq (m+1)\lambda^* \tag{11}$$

and

$$x_0=0,$$

where $L_{i_m}^{-1}$ is the inverse function of L_{i_m}.

These constraints show how the state variable, total target damage, changes from stage to stage. They simply indicate, at stage m when the damage is x_m and the decision is i_m, that to compute the damage at the beginning of stage $m+1$ the L_{i_m} function must be entered at the point x_m and then evaluated one more unit to the right. For feasibility the x_{m+1} obtained must lie below the λ^* line whose value at $r=m+1$ is $(m+1)\lambda^*$.

This single state variable dynamic programming formulation can be used to compute the optimal firing doctrine at each target given λ^* and the functions $L_i(r)$.

REFERENCES

[1] Battle, C. T., "The Uniform AVDAM Deployment Method," Stanford Research Institute Technical Note SED TN 194 (Sept. 1967).

[2] Eckler, A. R. and S. A. Burr, "Mathematical Models of Target Coverage and Missile Allocation," Military Operations Research Society (1972).

[3] Everett, H., "Exhaustion of Interceptor Supplies," The Lambda Corporation Paper 17, Defense Models XI (Apr. 1968).

LANCHESTER-TYPE MODELS OF WARFARE AND OPTIMAL CONTROL*

James G. Taylor

Department of Operations Research
and Administrative Sciences
Naval Postgraduate School
Monterey, California

ABSTRACT

The optimization of the dynamics of combat (optimal distribution of fire over enemy target types) is studied through a sequence of idealized models by use of the mathematical theory of optimal control. The models are for combat over a period of time described by Lanchester-type equations with a choice of tactics available to one side and subject to change with time. The structure of optimal fire distribution policies is discussed with reference to the influence of combatant objectives, termination conditions of the conflict, type of attrition process, and variable attrition-rate coefficients. Implications for intelligence, command and control systems, and human decision making are pointed out. The use of such optimal control models for guiding extensions to differential games is discussed.

1. INTRODUCTION

In this paper the structure of optimal fire distribution policies is examined for tactical situations described by Lanchester-type equations of warfare. This is done to provide insight into such important questions as

(1) How should fire be distributed over targets?
(2) Do target priorities change with time?
(3) Does the number of target types affect the optimal distribution of fire?
(4) Do battle termination circumstances affect the optimal allocation policies?
(5) How does the nature of the attrition process affect the optimal distribution of fire?
(6) How does the uncertainty and confusion of combat affect the optimal distribution rules?

A theory of tactical allocation is developed through the examination of a sequence of simplified models. These combat models are too simple to be taken literally but should be interpreted as indicating general principles to serve as hypotheses for subsequent computer simulation studies or field experimentation.

In 1964 Dolansky [9] noted that the Lanchester theory of combat was insufficiently developed in the area of target selection for combat between heterogeneous forces (optimal control/differential games). Even the two references cited by him, Weiss [31] and Isbell and Marlow [14], have been subsequently extended by this author [24], [26]. Since Dolansky's article, no further examples have been published in the literature except for the ones in Isaacs' book [13]. This previous work had never systematically investigated the dependence of optimal tactics upon model form.

*This research was supported by the Office of Naval Research as part of the Foundation Research Program at the Naval Postgraduate School.

Several idealized fire distribution problems are examined. The combat situations are described by Lanchester-type equations over a period of time with choices of tactics available to one side and subject to change over time. These problems are solved by the mathematical theory of optimal control. A further elaboration on solution development is to be found in the author's report [21].

The body of this paper is organized in the following fashion. First, a sequence of problems is considered, and the effect on the optimal distribution of fire is examined for the following factors: objectives of the combatants, termination conditions of the conflict, number of target types, some special cases of time dependent attrition-rate coefficients, and type of attrition process. Then, two-sided extensions of such problems are discussed, but the value of studying one-sided problems as considered in this paper is pointed out. Finally, various implications of the models are discussed.

2. NOTATION

The symbols which are used in this paper are defined as follows:

$a_1, \ldots, a_n, b_1, \ldots, b_n$ = constant attrition-rate coefficients,

$a_1(t), a_2(t), b_1(t), b_2(t)$ = variable attrition-rate coefficients,

$c_i(t)$ for $i=1, \ldots, n$ = coefficient of ϕ_i in maximization problem (defined by Equation (14)),

C_i for $i=1, 2, 3, 4, 5$ = the ith part of the terminal surface ("target set") as defined in section 3.2.,

$e_i(\tau)$ for $i=1, \ldots, n$ = coefficient of ϕ_i in maximization problem (defined by Equation (23)),

$h(t)$ = variable portion of variable attrition-rate coefficient, e.g., $a_1(t)=k_{a_1}h(t)$,

H = Hamiltonian function,

k = constant of proportionality,

$k_{a_1}, k_{a_2}, k_{b_1}, k_{b_2}$ = constant portions of variable attrition-rate coefficients, e.g., $a_1(t)=k_{a_1}h(t)$,

L = singular "surface" defined by $a_1b_1x_1=a_2b_2x_2$,

L' = line (with equation $a_1px_1=a_2qx_2$) in description of solution to Problem 5,

n = number of X-force target types,

$P^0=(x_1^0, \ldots, x_n^0, y_0)$ = point in the initial state space,

p, q, r = utilities assigned per unit of surviving X_1, X_2, and Y forces, respectively,

$p_i(t)$ for $i=1, \ldots, n+1$ = dual variable corresponding to $x_i(t)$ $(x_{n+1}(t)=y(t))$,

$R=a_1b_1/(a_2b_2)$,

R_i for $i=1, \ldots, n-1=a_i(b_iw_n-b_nw_i)/(a_ib_i-a_nb_n)$,

S_i for $i=1, \ldots, n=a_i(b_iw_k-b_kw_i)/(a_ib_i-a_kb_k)$,
$\quad\quad\quad

$\alpha = \frac{r}{q}\sqrt{\frac{b_2}{a_2}}$,

$\delta = a_1 p/(a_2 q)$,

δ_{ij} = Kronecker delta = $\begin{cases} 1 \text{ for } i=j, \\ 0 \text{ otherwise}, \end{cases}$

ϕ = fraction of Y-fire directed at X_1,

ϕ_i = fraction of Y-fire directed at X_i,

ϕ^* = optimal control,

τ = "backwards time" from the end of the battle; defined by $\tau = T - t$, i.e. the time remaining before the end of the battle,

τ_1, τ_2, etc. = "backwards time" of the first, second, etc., switch in tactics.

3. SOME FIRE DISTRIBUTION PROBLEMS

A theory of optimal fire distribution is developed by examining a sequence of problems and then contrasting the structures of the optimal fire distribution policies for these problems. In this manner the effect of the model's form on the optimal policy will be illustrated. Five different fire distribution problems are considered in order to study the effect of the model's form on the structure of the optimal policy by contrasting the solutions to these problems. The problems that are considered are for the optimal distribution of fire of a homogeneous force, denoted as Y, in Lanchester combat against heterogeneous forces, denoted as X_1 through X_n. These problems are summarized in Table I. The effects of the following four factors on the optimal allocation policy may be inferred from this study: number of target types, target-type attrition process, time variations in attrition-rate coefficients, and battle termination conditions.

TABLE I. *Summary of Problems Considered to Study Effect of Model Form on Optimal Fire Distribution Policy*

Problem	Number of Target Types	Target-type Attrition Process	Attrition-Rate Coefficients	Battle Termination Conditions
1	2	S	C	PD
2	2	S	C	TC
3	n	S	C	PD
4	2	S	V	PD
5	2	L	C	PD

Explanation of Symbols

Target-type Attrition Process

L = "linear-law" attrition process = attrition rate proportional to product of numbers of firers and targets

S = "square-law" attrition process = attrition rate proportional to only number of firers

Attrition-Rate Coefficients

C = constant

V = variable

Battle Termination Conditions

PD = prescribed duration battle (special case of $x_1 > 0$, $x_2 > 0$, $y > 0$)

TC = terminal control battle (fight-to-the-finish)

3.1. Battle of Prescribed Duration (Two Target Types)

The simplest fire distribution problem is for combat between an X-force of two force types (for example, riflemen and grenadiers) and a homogeneous Y-force (for example, riflemen only). This situation is shown diagrammatically in Figure 1. It is the objective of the Y-force commander to maximize his survivors at the end of battle at time T and minimize those of his opponent (considering weighting factors p, q, and r). This is accomplished through his choice of the fraction of fire, ϕ, directed at X_1. However, this idealized tactical allocation problem may be studied in two different scenarios: (1) a battle lasting a specified length of time (denoted as T_1) or (2) a battle lasting until one side or the other is totally annihilated. Each of these situations will be analyzed separately.

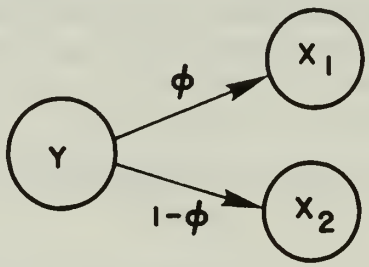


FIGURE 1. *Diagram of fire distribution problem*

Thus, Problem 1 is a prescribed duration battle. (The reader should recall that the definitions of Problems 1 through 5 are given in Table I.) It is stated in mathematical terms below.

$$\text{(Problem 1)} \qquad \underset{\phi(t)}{\text{maximize}} \{ry(T) - px_1(T) - qx_2(T)\} \text{ with } T_1 \text{ specified,}$$

$$\text{subject to:} \quad \frac{dx_1}{dt} = -\phi a_1 y,$$

$$\frac{dx_2}{dt} = -(1-\phi)a_2 y,$$

$$\frac{dy}{dt} = -b_1 x_1 - b_2 x_2,$$

$$x_1, x_2, y \geq 0, \qquad 0 \leq \phi \leq 1, \qquad \text{and } T \leq T_1,$$

where all symbols are defined in section 2. above.

3.1.1. Optimal Policy in a Special Case

The battle lasts for $0 \leq t \leq T_1$ unless, of course, one side or the other is annihilated before T_1. To be more precise, the battle terminates under one of the following three conditions:

(1) $x_1(T)=x_2(T)=0$ and $T \leq T_1$,

(2) $y(T)=0$ and $T \leq T_1$,

(3) $T=T_1$,

where T denotes the time at which the battle ends. However, to avoid inessential complications only the special case in which $x_1(T)>0$, $x_2(T)>0$, $y(T)>0$, and $T=T_1$ is considered for the comparisons made in this paper. In other words, those subcases in which a state variable is reduced to zero are not considered.* Thus, it is assumed that the initial force levels are such that no force type is annihilated during this prescribed duration battle.

The solution to Problem 1 for the above special case is shown in Table II. A derivation of these results is omitted, since Problem 1 may be considered to be a special case of Problem 3 for which a derivation is provided.

TABLE II. *Solution to Fire Distribution Problem (Problem 1) Battle of Prescribed Duration with Constant Attrition-Rate Coefficients; Special Case in which* $x_1(T)>0$, $x_2(T)>0$, $y(T)>0$

(Nonrestrictive assumption: $a_1b_1 > a_2b_2$)

Case	Optimal Control
A: $a_1p \geq a_2q$	$\phi^*(t)=1$ for $0 \leq t \leq T$
B: $a_1p < a_2q$	(a) for $\tau_1 \geq T$ $\phi^*(t)=0$ for $0 \leq t \leq T$
	(b) for $\tau_1 < T$ $\phi^*(t)=\begin{cases}1 \text{ for } 0 \leq t \leq T-\tau_1 \\ 0 \text{ for } T-\tau_1 \leq t \leq T\end{cases}$

NOTE: The "backwards" switching time is given by

$$\tau_1=\frac{1}{\sqrt{a_2b_2}}\ln\left\{\frac{z+\sqrt{z^2+\alpha^2-1}}{1+\alpha}\right\},$$

where $z=\dfrac{R-\delta}{R-1}$, $\delta=a_1p/(a_2q)$, $R=a_1b_1/(a_2b_2)$, and $\alpha=\dfrac{r}{q}\sqrt{\dfrac{b_2}{a_2}}$.

3.1.2. Discussion of Structure of Optimal Policy

With reference to Table II, two characteristics of the optimal allocation policies for this particular prescribed duration battle are:

(1) all fire is always concentrated on one target type;

(2) the allocation is not (directly) dependent upon the force levels.

---

*In the above problem x_1, x_2, and y are called *state* variables, while ϕ is called a *control* variable. A constraint such as $x_1 \geq 0$ is called a state variable inequality constraint (SVIC) and requires special treatment (see chapter 6 of [19]). Moreover, McIntyre and Paiewonsky [18] remarked in 1967 that "the optimal control problem with state space constraints does not appear to be well understood." The personal experience of this author bears this out [23].

It will be seen later that when there are more than two target types in this scenario, the solution possesses these same characteristics (even when the attrition rates change over time). Both these characteristics, however, are consequences of the assumed model form.

The first characteristic, concentration of effort on one alternative, is a consequence of the "square-law" attrition process for the X-forces. (The attrition of a target type will be referred to as being a "square-law" process when the casualty rate is proportional to the number of enemy firers only and as being a "linear-law" process when it is proportional to the product of the numbers of enemy firers and remaining targets.) It will be shown in section 3.3.2. that this makes the existence of a singular control [15] impossible, and hence the optimal allocation policies are extreme points in the control variable space.

There is, however, a very simple principle which underlies the above mathematical formalities: concentration of effort when constant marginal returns are obtained from the alternatives and the total effort is limited. Constant marginal effect over time per unit of weapon system is a property of the "square-law" attrition process; this is readily seen from considering the X_1-force attrition (when $\phi=1$)

$$\frac{\left(-\frac{dx_1}{dt}\right)}{y} = a_1 = \begin{pmatrix}\text{rate of casualties produced per}\\ \text{unit of } Y\text{-force weapon system}\end{pmatrix}.$$

Thus there is a constant (or nondiminishing) marginal effect over time. This should be contrasted with the situation for a "linear-law" attrition of the X_1-forces,

$$\frac{\left(-\frac{dx_1}{dt}\right)}{y} = a_1 x_1 = \begin{pmatrix}\text{rate of casualties produced per}\\ \text{unit of } Y\text{-force weapon system}\end{pmatrix}.$$

In this there are diminishing effects over time from allocating a unit of Y-force weapon system against X_1, and a division of total effort (i.e., fraction of fire) may be called for. B. Koopman's 1953 article [17] contains an excellent discussion of such principles which underlie such an optimization problem. Presently, these heuristic arguments will be verified in a mathematically precise fashion when a dynamic model, which considers the interaction of forces over time and in which both X-force target types undergo "linear-law" attrition, is considered. This fundamental difference in the structure of optimal allocation policies based on the nature of target attrition makes the selection of the appropriate attrition process an essential task of analysis.

The second characteristic, the optimal allocation not (directly) dependent upon the force levels, is due to the combination of the "square-law" attrition process for the X-force types and the fixed battle length. T. It is seen that for the special case of this prescribed duration battle in which $x_1(T) > 0$, $x_2(T) > 0$, and $y(T) > 0$ the optimal distribution of fire depends only on the attrition rates of the various force types and relative weights assigned to surviving force types. This is not surprising, since the adjoint differential equations (see section 3.3.2. below) are independent of the state variables, and the values of the dual variables at the end of battle $t=T$ are independent of force strengths. It is recalled that a dual variable represents the rate of change of the payoff (battle outcome as measured by the value of surviving forces at $t=T$) with respect to a particular state variable [2].

It seems appropriate to discuss further the interpretation of the solution shown in Table II. From the above definition of the dual variables,

$$a_1 p_1(t) = \begin{pmatrix}\text{effect on outcome per unit}\\ \text{time for engaging } X_1\end{pmatrix} = \begin{pmatrix}\text{kill rate of}\\ Y \text{ against } X_1\end{pmatrix} \times \begin{pmatrix}\text{effect on outcome per}\\ \text{unit of } X_1 \text{ destroyed}\end{pmatrix}.$$

Hence, the condition $a_1 p < a_2 q$ means that at the end of the battle (recall that $p_1(t=T) = -p$, etc.) there is greater effect on battle outcome (as measured by value of survivors) per unit time per soldier for Y to engage X_2 (short term gain at the end of battle). The value of the dual variable, for example, $p_1(t)$ reflects both the value assigned X_1-force survivors and the dynamic interaction of forces over time through the Lanchester-type equations. Hence, it also accounts for the effectiveness of X_1 against Y. The quantity $a_1 b_1$ may be interpreted as representing the instantaneous rate of destruction of the X_1-force kill rate against the Y-force per unit of Y-force. Then $a_1 b_1 > a_2 b_2$ means that there is greater strategic value for engaging the X_1-force, i.e., more long range return. Thus, Case A of Table II corresponds to when there is both more long range and also short range return for engaging X_1. Case B corresponds to when there is more short term gain at the end of the battle for engaging X_2, but more long range return for engaging X_1. It is easily shown that Case A results when Y values surviving X_1-forces greater than or equal to in direct proportion to their kill rate against the Y-force, i.e., $p/q \geq b_1/b_2$. A switch in tactics (target priority) is seen to occur for this model only when value is not assigned to X_1 survivors (recall that engagement of X_1 always yields more "long range return") greater than or equal to in proportion to their destructive capability (kill rate).

The maximum principle may be interpreted as saying that a target type from several alternatives is engaged when such an engagement yields the greatest favorable effect on battle outcome per unit time. It turns out, though, that the evolution of target engagement return is dependent upon the scenario chosen for the study of the problem. This is clearly seen when we examine the "fight-to-the-finish." This is a special case of a terminal control battle (the combat ends only when the course of battle has been steered to a prescribed end state) and is chosen for mathematical convenience.

3.2. Terminal Control Battle (Two Target Types)

Problem 2 is a terminal control battle (a "fight-to-the-finish") and is stated in mathematical terms below.

(Problem 2)

$$\underset{\phi(t)}{\text{maximize}} \{ry(T) - px_1(T) - qx_2(T)\} \text{ with } T \text{ unspecified,}$$

$$\text{subject to:} \quad \frac{dx_1}{dt} = -\phi a_1 y,$$

$$\frac{dx_2}{dt} = -(1-\phi) a_2 y,$$

$$\frac{dy}{dt} = -b_1 x_1 - b_2 x_2,$$

$$x_1, x_2, y \geq 0 \text{ and } 0 \leq \phi \leq 1,$$

where all symbols are defined in section 2. above.

The stopping rule for this battle is that the conflict terminates at $t=T$ defined by

(a) $y(T)=0$,
or (*b*) $x_1(T)=x_2(T)=0$.

Upon further analysis it has been convenient to consider that there are the following five "target sets" for Problem 2:

$$C_1\colon\ x_1(T)=0,\ x_2(T)>0,\ y(T)=0,$$
$$C_2\colon\ x_1(T)=0 \text{ before } x_2(T)=0,\ y(T)>0,$$
$$C_3\colon\ x_1(T)=0 \text{ after } x_2(T)=0,\ y(T)>0,$$
$$C_4\colon\ x_1(T)>0,\ x_2(T)=0,\ y(T)=0,$$
$$C_5\colon\ x_1(T)>0,\ x_2(T)>0,\ y(T)=0.$$

The reader should note that in the above problem statement T is referred to as being undetermined. This is because T is determined by entry to one of the above five target sets. In turn, this depends upon the control used, and hence before an allocation rule is given, it is unspecified.

Problem 2 was first studied by Isbell and Marlow [14] in 1956. However, their solution was not correct for all values of model parameters, since they did not discover the dispersal surfaces (see pp. 132–141 of [13]) present in this problem's solution for a certain range of model parameters. A more complete solution has been given by the author in a previous paper [24]. The solution principles for solving such an optimal control problem may be extended to the special class of terminal control differential games which have pure strategy solutions. This was done by the author in [26] and used to solve the supporting weapon system game of H. K. Weiss [31].

In describing the solution to Problem 2 (see [24] for the details of its development) three cases must be considered

(1) $\delta \geq 1$,
(2) $R-\sqrt{R(R-1)} \leq \delta < 1$,
(3) $0 \leq \delta < R-\sqrt{R(R-1)}$,

where $\delta = a_1p/(a_2q)$. The solution for each of these cases is given in [24]. Moreover, these appearingly complex results may be summarized in a particularly simple fashion* (for the nonrestrictive assumption that $R>1$, i.e., $a_1b_1 > a_2b_2$). When Y wins, he engages X_1 until depletion before X_2. When Y loses, he may switch from firing at X_1 entirely to firing at X_2 entirely before the X_1 force has been annihilated. This happens in Case (2) ($R-\sqrt{R(R-1)} \leq \delta < 1$) when survivors of force-type X_2 are assigned utility in excess of their Lanchester attrition-rate coefficient as compared with force-type X_1, and certain relationships hold between initial force strengths. Thus, in contrast to the prescribed duration battle (Problem 1), the optimal policy for Problem 2 may depend on initial force levels.

Finally, it seems appropriate to point out that the "backwards" switching time, denoted as τ_1, is different in these two problems. Let τ_1 (Problem 1) denote the "backwards" switching time for an optimal policy in Problem 1. It represents the optimal length of time that Y fires at X_2 before the end o

*Thus, for comparison purposes of the present paper the complete solution need not be given here.

battle at $t=T$ when $a_1p < a_2q$ (for the special case in which $x_1(T) > 0$, $x_2(T) > 0$, $y(T) > 0$, and $\tau_1 \leq T$). It is convenient to define

$$\tau_1(a) = \frac{1}{\sqrt{a_2 b_2}} \ln \frac{z + \sqrt{z^2 + a^2 - 1}}{1 + a}. \tag{1}$$

From Table II it is seen that

$$\tau_1 \text{ (Problem 1)} = \tau_1\left(a = \frac{r}{q}\sqrt{\frac{b_2}{a_2}}\right). \tag{2}$$

It is assumed that r is a strictly positive quantity. Furthermore, in Problem 2 τ_1 denotes the backwards time at which ϕ^* changes from 0 to 1 (in backwards progression) with $x_1(\tau = \tau_1) > 0$, i.e., the time of a change in the optimal distribution of fire without the annihilation of a target type. In [24] it was shown that

$$\tau_1 \text{ (Problem 2)} = \tau_1(a = 0). \tag{3}$$

Then, it is easy to show that when $\delta = a_1p/(a_2q) < 1$, it follows that τ_1 (Problem 1) $< \tau_1$ (Problem 2) and in such a case Y fires at X_2 for a longer period of time in Problem 2 than in Problem 1. This is stated as Theorem 1.

THEOREM 1: Assume that $\delta < 1$ and $r > 0$. Then

$$\tau_1 \text{ (Problem 1)} < \tau_1 \text{ (Problem 2)}.$$

By observing (2) and (3) and recalling that $q, r > 0$, we can see that the theorem follows by showing that $\frac{\partial \tau_1}{\partial a} < 0$ for $\delta < 1$. It is readily computed from (1) that

$$\frac{\partial \tau_1}{\partial a} = \frac{a + 1 - z^2 - z\sqrt{z^2 + a^2 - 1}}{\sqrt{a_2 b_2}(1 + a)\sqrt{z^2 + a^2 - 1}(z + \sqrt{z^2 + a^2 - 1})}. \tag{4}$$

Now $\delta < 1$ (recalling the nonrestrictive assumption $R > 1$) implies that $z > 1$, so that

$$a < za < z\sqrt{z^2 + a^2 - 1}, \tag{5}$$

since $a > 0$. From (5) it follows that

$$a + 1 - z^2 - z\sqrt{z^2 + a^2 - 1} < o, \tag{6}$$

which proves that $\frac{\partial \tau_1}{\partial a} < 0$ for $\delta < 1$.

3.3. Prescribed Duration Battle with Several Target Types

The first two problems were considered in order to contrast the structures of the optimal allocation policies for different battle termination conditions. Another factor that can be examined is the number of target types. For the prescribed duration battle certain facets which tended to be obscured in the scenario with two target types are brought into sharper focus when n target types are considered. Thus, Problem 3 is a prescribed duration battle against n target types and is stated in mathematical terms below.

(Problem 3)

$$\underset{\phi_i(t)}{\text{maximize}} \{vy(T) - \sum_{i=1}^{n} w_i x_i(T)\} \text{ with } T_1 \text{ specified,}$$

$$\text{subject to: } \frac{dx_i}{dt} = -\phi_i a_i y \text{ for } i = 1, \ldots, n,$$

$$\frac{dy}{dt} = -\sum_{i=1}^{n} b_i x_i,$$

$$x_i, y \geq 0, \phi_i \geq 0 \text{ for } i = 1, \ldots, n \text{ and } \sum_{i=1}^{n} \phi_i = 1.$$

3.3.1. Optimal Policy in a Special Case

The battle lasts for $0 \leq t \leq T_1$ unless, of course, one side or the other is annihilated before T_1. To be more precise, the battle terminates under one of the following three conditions:

(1) $x_1(T) = \ldots = x_n(T) = 0$ and $T \leq T_1$,

(2) $y(T) = 0$ and $T \leq T_1$,

(3) $T = T_1$,

where T denotes the time at which the battle ends. However, for the comparisons made in this paper, only the special case in which $x_1(T) > 0, \ldots x_n(T) > 0, y(T) > 0$, and $T = T_1$ is considered. Thus, as done in section 3.1.1. it is assumed that the initial force levels are such that no force type is annihilated during this prescribed duration battle.

The solution to Problem 3 for the above special case is shown in Table III. A derivation of these results is given in the next section. In Table III δ_{ij} denotes the Kronecker delta and is equal to 1 for $i = j$ and zero otherwise. It is seen that the solution to Problem 3 turns out to be a generalization of that to Problem 1. However, certain aspects receive greater emphasis to provide one with a deeper understanding of the phenomena under study. In particlular, two subcases, denoted as Case A and Case B, were considered in the solution to Problem 1 (see Table II). When there are several target types, the generalization of the subcases, which it is convenient to distinguish, is as follows:

Case A, enemy survivors valued in direct proportion to their kill rate against Y-force,

Case B, enemy survivors *not* valued in direct proportion to their kill rate against Y-force.

In the first instance, Case A, target priorities keep their same relative ranking over time. If the highest priority target type is exterminated during such a battle, then fire is merely shifted to the next highest priority target. Hence, when one values enemy survivors in proportion to their kill rate against you, i.e., $w_i = kb_i$, for $i = 1, \ldots, n$, the optimal tactic is to concentrate all fire on a single target type until it is entirely destroyed. The sole criterion for target selection in this instance is the quantity $a_i b_i$, which may be interpreted to be the rate of destruction of enemy attrition capability for his ith force type (see section 3.1.2).

TABLE III. *Solution to Fire Distribution Problem (Problem 3) Battle of Prescribed Duration with* Constant Attrition Rates; Special Case in Which $x_1(T) > 0$ *for* $i = 1, \ldots, n$ *and* $y(T) > 0$

Case	Optimal Control
A: $w_i = kb_i$ for $i = 1, \ldots, n$	$\phi_i^*(t) = \delta_{ij}$ for $0 \leq t \leq T$ $i = 1, \ldots, n$
B: $w_i \neq kb_i$ for at least one index i	(a) for $\tau_1 \geq T$ $\phi_i^*(t) = \delta_{in}$ for $0 \leq t \leq T$ $i = 1, \ldots n$
	(b) for $\tau_2 \geq T > \tau_1$ $\phi_i^*(t) = \begin{cases} \delta_{ik} \text{ for } 0 \leq t \leq T - \tau_1 \\ \delta_{in} \text{ for } T - \tau_1 < t \leq T \end{cases}$ $i = 1, \ldots, n$
	(c) for $\tau_3 \geq T > \tau_2$ $\phi_i^*(t) = \begin{cases} \delta_{ij} \text{ for } 0 \leq t \leq T - \tau_2 \\ \delta_{ik} \text{ for } T - \tau_2 < t \leq T - \tau_1 \\ \delta_{in} \text{ for } T - \tau_1 < t \leq T \end{cases}$ $i = 1, \ldots, n$
	etc.

NOTES:

(1) J is index such that $a_J b_J = \max(a_1 b_1, \ldots, a_n b_n)$.

(2) n is index assigned so that $a_n w_n = \max(a_1 w_1, \ldots, a_n w_n)$.

(3) k is index such that $R_k = \min_{\substack{R_i > 0 \\ a_i b_i > a_n b_n}} (R_1, \ldots, R_{n-1})$ where $R_i = \dfrac{a_i(b_i w_n - b_n w_i)}{a_i b_i - a_n n_n}$ for $i = 1, \ldots, n-1$.

(4) τ_1 is given by $\tau_1 = \dfrac{1}{\sqrt{a_n b_n}} \ln \left\{ \dfrac{\left(\dfrac{R_k}{w_n}\right) + \sqrt{\left(\dfrac{R_k}{w_n}\right)^2 + \left(\dfrac{v}{w_n}\right)^2 \left(\dfrac{b_n}{a_n}\right) - 1}}{1 + \dfrac{v}{w_n}\sqrt{\dfrac{b_n}{a_n}}} \right\}$.

(5) j is index such that $S_j = \min_{\substack{S_i > 0 \\ a_i b_i > a_k b_k \\ i \neq k}} (S_1, \ldots, S_n)$ where $S_i = \dfrac{a_i(b_i w_k - b_k w_i)}{a_i b_i - a_k b_k}$ for $i = 1, \ldots, n$. $_{i \neq k}$

(6) τ_2 is given by $\tau_2 = \tau_1 + \dfrac{1}{\sqrt{a_k b_k}} \ln \left\{ \dfrac{\left(\dfrac{S_j}{W_k}\right) + \sqrt{\left(\dfrac{S_j}{W_k}\right)^2 + \left(\dfrac{V_k}{W_k}\right)^2 \left(\dfrac{b_k}{a_k}\right) - 1}}{1 + \dfrac{V_k}{W_k}\sqrt{\dfrac{b_k}{a_k}}} \right\}$.

(7) τ_3 is given by expression similar to those for τ_1 and τ_2 above.

With reference to Table III in Case B it is seen that there may be one or more switches in target priorities if the battle lasts long enough. For example, in subcase (b) of Case B of Table III a switch in the optimal tactic of concentrating all fire on one target type occurs, and fire is shifted from target type k to target type n. It will be shown that necessary conditions for fire to be switched from target type k to n are that $a_k b_k > a_n b_n$ and $\dfrac{b_k}{b_n} > \dfrac{w_k}{w_n}$, i.e., fire is shifted from a target type which causes attrition in a greater proportion than the ratio of values placed upon survivors to the target type which yields the greatest direct return at the end of battle. Additionally, in Table III explicit expressions are given for "switching times" as well as for the determination of the target type upon which all fire is concentrated.

It should be noted that when $n=2$ the results of Table III reduce to those given in Table II. To see this one sets $n=2$ and makes the following identifications: w_1 in Problem 3 is replaced by p in Problem 1, and w_2 by q.

3.3.2. Development of Optimal Fire Distribution Policy

For $y>0$ and $x_i>0$ for $i=1, \ldots, n$, the Hamiltonian for Problem 3 is given by [8]*

$$H(t, x_i, p_i, \phi_i) = -y \sum_{i=1}^{n} a_i p_i(t) \phi_i - p_{n+1} \sum_{i=1}^{n} b_i x_i, \tag{7}$$

where $p_i(t)$ for $i=1, \ldots, n$ denotes the dual variable corresponding to x_i and $p_{n+1}(t)$ denotes the dual variable corresponding to y. According to the maximum principle, the optimal control (there is only one extremal) is determined by the (trivial) linear program

$$\begin{aligned} &\underset{\phi_i}{\text{maximize}}\ H(t, x_i, p_i, \phi_i) \\ &\text{subject to: } \sum_{i=1}^{n} \phi_i = 1, \\ &\phi_i \geq 0, \end{aligned}$$

which in turn leads to

$$\begin{aligned} &\underset{\phi_i}{\text{maximize}} \sum_{i=1}^{n} a_i(-p_i(t))\phi_i \\ &\text{subject to: } \sum_{i=1}^{n} \phi_i = 1, \\ &\phi_i \geq 0. \end{aligned} \tag{8}$$

By inspection the solution to (8) is easily seen to be

$$\phi_i^*(t) = \delta_{ij}(t), \tag{9}$$

where δ_{ij} is the Kronecker delta and is equal to 1 for $i=j$ and zero otherwise and $j(t)$ is the index such that

$$a_j p_j(t) = \text{minimum } (a_1 p_1, a_2 p_2, \ldots, a_n p_n).$$

To trace the history of ϕ_i^* over time, one must consider the adjoint system of differential equations given by

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial x_i} = p_{n+1} b_i \text{ with } p_i(t=T) = -w_i \text{ for } i=1, \ldots, n,$$

*There is a difference in sign between the version of the maximum principle used by Pontryagin et al. [19] and an equivalent version commonly used in the control theory literature of this country (see p. 108 of [8]).

and

$$\frac{dp_{n+1}}{dt}=-\frac{\partial H}{\partial y}=\sum_{i=1}^{n}\phi_i^* a_i p_i \text{ with } p_{n+1}(t=T)=v. \tag{10}$$

It may be that the index $j(t)$ is not unique, i.e., the linear program (8) has alternate optima. This causes no difficulty unless this situation continues for a finite interval of time. When this happens, the corresponding segment of the battle trajectory is called a *singular subarc* [15]. However, it is easily shown that it is impossible to have a singular solution of Problem 3. If $j(t)$ were not unique for a finite interval of time, then (for example) one would have $a_j p_j(t)=a_k p_k(t)$ for $t_1 \leq t \leq t_2$. If this were to occur, then one must have

$$a_j \frac{dp_j}{dt}=a_k \frac{dp_k}{dt},$$

or using (10)

$$p_{n+1}(t)\,(a_j b_j - a_k b_k)=0. \tag{11}$$

Since $p_{n+1}(t)>0$ for $0\leq t \leq T$, Equation (11) implies that $a_j b_j = a_k b_k$, which, in general, is not true. Hence, there is no singular solution to Problem 3 and $\phi_i^*(t)$ is either 0 or 1 (almost everywhere).

Considering (10), it is easily seen that

$$\frac{dp_i}{dp_n}=\frac{b_i}{b_n},$$

so that

$$p_i(t)=\frac{b_i}{b_n}\{p_n(t)+w_n\}-w_i. \tag{12}$$

Substituting (12) into (8), one obtains after some manipulation that the optimal control is determined by

$$\begin{aligned} &\underset{\phi_i}{\text{maximize}} \sum_{i=1}^{n} c_i(t)\phi_i \\ &\text{subject to: } \sum_{i=1}^{n}\phi_i=1, \\ &\qquad\qquad \phi_i \geq 0, \end{aligned} \tag{13}$$

where

$$c_i(t)=\frac{(-p_n(t))}{b_n}\,a_i b_i\left[1+\frac{w_i}{(-p_n(t))}\left\{\frac{b_n}{b_i}-\frac{w_n}{w_i}\right\}\right]. \tag{14}$$

Hence, it is seen that if $w_i=kb_i$ for $i=1, \ldots, n$, i.e., Y values enemy survivors in direct proportion to their kill rate against his forces, the above problem is equivalent to

$$\begin{aligned} &\underset{\phi_i}{\text{maximize}} \sum_{i=1}^{n}\phi_i a_i b_i \\ &\text{subject to: } \sum_{i=1}^{n}\phi_i=1, \\ &\qquad\qquad \phi_i \geq 0, \end{aligned}$$

where use has been made of the easily verified fact that $p_n(t) < 0$ for all time. Thus the optimal control is given by

$$\phi_i^*(t) = \delta_{iJ} \text{ for } 0 \leq t \leq T, \tag{15}$$

where J is the index such that

$$a_J b_J = \text{maximum } (a_1 b_1, \ldots, a_n b_n).$$

Hence, for this problem when one values enemy survivors in direct proportion to their kill rate against you, the optimal tactic is to concentrate all fire on a single target type until it is entirely destroyed. The result (15) is given in Table III.

The more complex case in which one does not value enemy survivors in direct proportion to their kill capability (measured by a Lanchester attrition-rate coefficient) against you will now be considered. Since the solution to this problem is developed by working backwards from the end $t=T$, it is convenient to introduce the "backwards time" variable τ defined by $\tau = T - t$. It is assumed that the enemy target types have been indexed so that n is the index such that

$$a_n w_n = a_n(-p_n(\tau=0)) = \text{maximum } (a_1 w_1, \ldots, a_n w_n). \tag{16}$$

By (8) it is easily seen that

$$\phi_i^*(t=T) = \delta_{in}. \tag{17}$$

By straightforward continuity arguments, it is readily seen that

$$\phi_i^*(\tau) = \delta_{in} \text{ for } \tau \epsilon [0, \tau_1), \tag{18}$$

where τ_1 is the "backwards time" of the first switch in target selection. Giving consideration to (18) and observing that $\frac{d}{dt} = -\frac{d}{d\tau}$, it is seen that for $\tau \epsilon [0, \tau_1]$ one need only consider the following equations from the adjoint system (10)

$$\frac{dp_n}{d\tau} = -b_n p_{n+1} \text{ with } p_n(\tau=0) = -w_n,$$

$$\frac{dp_{n+1}}{d\tau} = -a_n p_n \text{ with } p_{n+1}(\tau=0) = v, \tag{19}$$

and it is recalled that (rewriting (12))

$$p_i(\tau) = \frac{b_i}{b_n}\{p_n(\tau) + w_n\} - w_i \text{ for } i = 1, \ldots, n. \tag{20}$$

The above initial value problem (19) is routinely solved to yield

$$p_n(\tau) = -w_n \cosh \sqrt{a_n b_n}\,\tau - v\sqrt{\frac{b_n}{a_n}} \sinh \sqrt{a_n b_n}\,\tau. \tag{21}$$

It will now be determined what conditions are necessary for a change in target selection and the time at which the change occurs, τ_1. To do this it is convenient to rewrite (13) and (14) as

$$\underset{\phi_i}{\text{maximize}} \sum_{i=1}^{n} e_i(\tau)\phi_i$$

$$\text{subject to: } \sum_{i=1}^{n} \phi_i = 1,$$

$$\phi_i \geq 0, \tag{22}$$

where

$$e_i(\tau) = a_i w_i \left[1 + \frac{b_i}{w_i b_n} \{(-p_n(\tau)) - w_n\}\right]. \tag{23}$$

A switch in the optimal distribution of fire occurs at the smallest τ for which

$$a_i w_i \left[1 + \frac{b_i}{w_i b_n} \{(-p_n(\tau)) - w_n\}\right] = a_n(-p_n(\tau)), \tag{24}$$

where $i = 1, \ldots, n-1$ and certain other conditions (to be determined presently) are met. Let k be the index of the target type to which fire is first shifted in "backwards time." Observe that at $\tau = 0$ one has

$$a_i w_i < a_n w_n, \tag{25}$$

for $i = 1, \ldots, n-1$, since the index n has been defined by (16). Then for $\tau_1 < \tau < \tau_2$, where τ_2 is the "backwards time" of the second switch in target selection, one has that $\phi_i^*(\tau) = \delta_{ik}$, and thus by (22) and (23) the following inequality must hold

$$a_k w_k \left[1 + \frac{b_k}{w_k b_n} \{(-p_m(\tau)) - w_n\}\right] > a_n(-p_n(\tau)),$$

which may be rearranged to yield

$$a_k(b_k w_n - b_n w_k) < (a_k b_k - a_n b_n)(-p_n(\tau)). \tag{26}$$

It will now be shown that a necessary condition for fire to be shifted from target type n to target type k when one works backwards from the end is that $a_k b_k > a_n b_n$. The proof is as follows. It will be

shown that $a_k b_k \leqslant a_n b_n$ leads to a contradiction. First, consider the special case when $a_k b_k = a_n b_n$. In this case (26) reduces to

$$\frac{w_n}{w_k} < \frac{b_n}{b_k} = \frac{a_k}{a_n},$$

or

$$a_n w_n < a_k w_k.$$

But this is a contradiction to (25) which must hold with $i = k$. In the case when $a_n b_n > a_k b_k$, then using the fact that $(-p_n(\tau)) > w_n$ for $\tau > 0$, we may write (26) as

$$\frac{a_k(b_n w_k - b_k w_n)}{(a_n b_n - a_k b_k)} > (-p_n(\tau)) > w_n,$$

but this leads to $a_k w_k > a_n w_n$ which is a contradiction to (25) as before.

Thus, $a_k b_k > a_n b_n$ and the switch in target selection occurs at

$$\frac{a_k(b_k w_n - b_n w_k)}{(a_k b_k - a_n b_n)} = (-p_n(\tau = \tau_1)) > 0, \tag{27}$$

so that a second necessary condition is

$$\frac{b_k}{b_n} > \frac{w_k}{w_n}.$$

In other words, all fire is concentrated at earlier (forward) times in the battle on the target type which causes attrition proportionally more than the ratio of values placed on survivors and then is switched later to the target type which yields the greatest direct return at the end of battle.

To recapitulate the above, the target to which fire is first shifted (working backwards from the end of battle) has index k determined by

$$R_k = \underset{\substack{R_i > 0 \\ a_i b_i > a_n b_n}}{\text{minimum}} (R_1, \ldots, R_{n-1}), \tag{28}$$

where

$$R_i = \frac{a_i(b_i w_n - b_n w_i)}{(a_i b_i - a_n b_n)} \qquad \text{for } i = 1, \ldots, n-1. \tag{29}$$

The time of switch, τ_1, of fire to the kth target type is determined by the equation

$$w_n \cosh \sqrt{a_n b_n}\, \tau_1 + v \sqrt{\frac{b_n}{a_n}} \sinh \sqrt{a_n b_n}\, \tau_1 = \frac{a_k(b_k w_n - b_n w_k)}{a_k b_k - a_n b_n}, \tag{30}$$

which may be solved to yield the expression for τ_1 given in Table III.

The general pattern of when and to which target types fire is shifted as one works backwards from the end of battle does not emerge until one has considered the second shift in target selection. Since this is dependent upon the evolution of target worth, the backwards integration of the adjoint system of differential equations must be further considered. From above, one has that

$$\phi_i^*(\tau) = \delta_{ik} \qquad \text{for } \tau \epsilon (\tau_1, \tau_2), \tag{31}$$

where τ_2 is the "backwards time" of the second switch in target selection. Giving consideration to (31), it is seen that for $\tau \epsilon [\tau_1, \tau_2]$ one needs only to consider the following equations from the adjoint system (10)

$$\frac{dp_k}{d\tau} = -b_k p_{n+1} \qquad \text{with } p_k(\tau = \tau_1) = -W_k,$$

$$\frac{dp_{n+1}}{d\tau} = -a_k p_k \qquad \text{with } p_{n+1}(\tau = \tau_1) = V_k, \tag{32}$$

where

$$W_k = \frac{a_n(b_k w_n - b_n w_k)}{a_k b_k - a_n b_n} \tag{33}$$

$$V_k = \sqrt{\frac{a_n}{b_n}(R_k^2 - w_n^2) + v^2}. \tag{34}$$

Equation (33) follows from the fact that by (8) and (9) at $\tau = \tau_1$ we have

$$a_k(p_k(-\tau = \tau_1)) = a_n(-p_n(\tau = \tau_1)), \tag{35}$$

which may be combined with (27), (28), and (29) to yield the desired result. Equation (34) is readily deduced by observing that according to (19) a "square law" relates the dual variables $p_n(\tau)$ and $p_{n+1}(\tau)$ for $0 \leq \tau \leq \tau_1$

$$a_n\{p_n^2(\tau) - w_n^2\} = b_n\{p_{n+1}^2(\tau) - v^2\}, \tag{36}$$

whence follows (34) by use of (27), (29), and (32). It should be noted that all the dual variables may be expressed in terms of $p_k(\tau)$ (let $n = k$ in (12))

$$p_i(\tau) = \frac{b_1}{b_k}\{p_k(\tau) + w_k\} - w_i \qquad \text{for } i = 1, \ldots, n. \tag{37}$$

Again, the Equations (32) are routinely solved to yield for $\tau \epsilon [\tau_1, \tau_2]$

$$p_k(\tau) = -W_k \cosh \sqrt{a_k b_k}\,(\tau - \tau_1) - V_k \sqrt{\frac{b_k}{a_k}} \sinh \sqrt{a_k b_k}\,(\tau - \tau_1). \tag{38}$$

Subsequent arguments are now similar to those given for the first switch in tactics. Let j be the index of the target type to which fire is shifted secondly in "backwards time." Then, it may be shown

by similar arguments to above that necessary conditions for fire to be shifted to the jth target type are that

$$a_j b_j > a_k b_k > a_n b_n, \tag{39}$$

and

$$\frac{b_j}{b_k} > \frac{w_j}{w_k}. \tag{40}$$

However, more insight may be gained by rewriting (40) as

$$\frac{b_j}{w_j} > \frac{b_k}{w_k} > \frac{b_n}{w_n} \tag{41}$$

It also seems appropriate to point out the military interpretation of the ratio $\frac{b_i}{w_i}$. Recall that

w_i = value per unit of X_i surviving at $t = T$,
b_i = kill rate per unit of X_i against Y.

Then

$$\frac{b_i}{w_i} = \frac{\text{kill rate per unit of } X_i}{\text{value per unit of } X_i \text{ survivors}}.$$

Thus, it is seen that as one progresses backwards from the end of battle that fire is always shifted to target types with larger ratios of kill rate per unit of weapon system per unit value of survivors.

Using an argument similar to the one used to develop (28) and (29) for the first shift in fire, it may be shown that the target to which fire is shifted secondly (working backwards from the end of battle) has index j determined by

$$S_j = \underset{\substack{S_i > 0 \\ a_i b_i > a_k b_k \\ i \neq k}}{\text{minimum}} (S_1, \ldots, S_n), \tag{42}$$

where

$$S_i = \frac{a_i(b_i w_k - b_k w_i)}{a_i b_i - a_k b_k} \qquad \text{for } i = \underset{i \neq k}{1, \ldots, n}. \tag{43}$$

The time of switch, τ_2, of fire to the jth target type is determined by the transcendental equation

$$W_k \cosh \sqrt{a_k b_k}\,(\tau_2 - \tau_1) + V_k \sqrt{\frac{b_k}{a_k}} \sinh \sqrt{a_k b_k}\,(\tau_2 - \tau_1) = \frac{a_j(b_j w_k - b_k w_j)}{a_j b_j - a_k b_k}, \tag{44}$$

which may be solved to yield the expression for τ_2 given in Table III. Further shifts in fire follow the pattern established above.

3.3.3. Discussion of Structure of Optimal Policy

Considering Table III, it is seen that the optimal allocation policy for Problem 3 has the same structure as that for Problem 1 with just two target types:

(1) fire is always concentrated on one target type,

(2) the allocation is not directly dependent upon the force levels.

The addition of more target types has not changed the nature of the problem: its explicit solution is a generalization of that with two X-force target types.

It is of interest to ask whether the optimal tactic will always be to concentrate fire on only one target type (bang-bang optimal control). The answer to this question turns out to be "no" as consideration of Problem 5 with a "linear-law" attrition process for the X-force target types will show. The reader is referred to section 3.1.2. for a further heuristic discussion of the structure of the optimal policy for the distribution of fire over target types which undergo attrition at a rate proportional to only the number of firers.

3.4. Some Special Cases of Time Dependent Attrition-Rate Coefficients

In the previous idealizations of combat that have been considered above, it has been assumed that all the Lanchester attrition-rate coefficients were constant. In reality, such a coefficient depends upon numerous factors some of which are as follows: hit probabilities, weapon system projectile-target lethality characteristics, rates of fire, rate of target acquisition. These factors themselves may be range dependent or change over time. S. Bonder [5], [6] has developed explicit formulas for relating the Lanchester attrition-rate coefficient to weapons system performance characteristics such as those mentioned above.

Thus, it seems appropriate to examine idealized combat situations in which the attrition-rate coefficients are time dependent. Moreover, this is facilitated by the author's research results on solutions to variable-coefficient Lanchester-type equations for "square-law" attrition processes [22], [27]. A key result is that there is a class of variable-coefficient Lanchester-type equations (combat between two homogeneous forces when the attrition-rate coefficents are variable provided that their ratio is constant) which possess a solution no more complicated than the solution to the constant coefficient case [22]. This type of property (reflecting the physical situation in which two weapon systems cause attrition in a proportional fashion at all times) will now be exploited in an optimal control problem.

Thus, Problem 4 is a prescribed duration battle and is stated in mathematical terms below.

(Problem 4) $\underset{\phi(t)}{\text{maximize}} \{ry(t) - px_1(T) - qx_2(T)\}$ with T_1 specified,

$$\text{subject to: } \frac{dx_1}{dt} = -\phi a_1(t) y,$$

$$\frac{dx_2}{dt} = -(1-\phi) a_2(t) y,$$

$$\frac{dy}{dt} = -b_1(t) x_1 - b_2(t) x_2,$$

$$x_1, x_2, y \geq 0, \qquad 0 \leq \phi \leq 1, \qquad \text{and } T \leq T_1.$$

It is assumed that both X-force weapon systems are such that

$$b_1(t) = k_{b_1} h(t) \qquad \text{and } b_2(t) = k_{b_2} h(t). \tag{45}$$

As done above for Problems 1 and 3, only the special case in which $x_1(T) > 0, x_2(T) > 0, y(T) > 0$, and $T = T_1$ is considered for the comparisons made in this paper. In the further special case in which the

TABLE IV. *Solution to Fire Distribution Problem (Problem 4) Battle of Prescribed Duration with Variable Attrition-Rate coefficients; Special Case in Which* $x_1(T) > 0$, $x_2(T) > 0$, $y(T) \geq 0$

Special assumption: $a_1(t)/a_2(t) = k_{a_1}/k_{a_2}$, $b_1(t)/b_2(t) = k_{b_1}/k_{b_2}$, and $a_1(t)/b_1(t) = k_{a_1}/k_{b_1}$
Nonrestrictive assumption: $k_{a_1}k_{b_1} > k_{a_2}k_{b_2}$

Case	Optimal Control
A: $k_{a_1}p \geq k_{a_2}q$	$\phi^*(t) = 1$ for $0 \leq t \leq T$
B: $k_{a_1}p < k_{a_2}q$	(a) for $\tau_1 \geq T$ $\phi^*(t) = 0$ for $0 \leq t \leq T$
	(b) for $\tau_1 < T$ $\phi^*(t) = \begin{cases} 1 & \text{for } 0 \leq t \leq T - \tau_1 \\ 0 & \text{for } T - \tau_1 \leq t \leq T \end{cases}$

NOTE: The "backwards" switching time is given by

$$\int_0^{\tau_1} h(\tau)\,d\tau = \frac{1}{\sqrt{k_{a_2}k_{b_2}}} \ln \frac{z + \sqrt{z^2 + \alpha^2 - 1}}{1 + \alpha},$$

where

$$z = \frac{R - \delta}{R - 1},\ \delta = k_{a_1}p/(k_{a_2}q),\ R = k_{a_1}k_{b_1}/(k_{a_2}k_{b_2}),\quad \text{and } \alpha = \frac{r}{q}\sqrt{\frac{k_{b_2}}{k_{a_2}}}.$$

Y-force values surviving X-force types in direct proportion to their kill rates (as measured by the Lanchester attrition-rate coefficients) against the Y-force, i.e., $p/q = k_{b_1}/k_{b_2} = b_1(t)/b_2(t) = b_1(t=T)/b_2(t=T)$, the optimal control law takes a particularly simple form

$$\phi^*(t) = \begin{cases} 1 & \text{for } a_1(t)b_1(t) > a_2(t)b_2(t), \\ 0 & \text{for } a_1(t)b_1(t) < a_2(t)b_2(t). \end{cases} \tag{46}$$

In this instance target selection depends only on the product of attrition-rate coefficients which may be interpreted as the rate of destruction of enemy kill-rate capability. All fire is concentrated on one of the target types depending on which target type has the larger product of attrition-rate coefficients. Target priority is subject to change over time as the ranking of the target types on this decision criterion changes. It is conceivable that the optimal tactic may be to shift fire from one target type to the other several times over the course of battle with the duration of battle not having any effect. Observe that no assumptions at all have been made on the Y-force attrition rates against X_1 and X_2, i.e., $a_1(t)$ and $a_2(t)$.

It is now further assumed that $a_1(t) = k_{a_1}h(t)$ and $a_2(t) = k_{a_2}h(t)$. This means that not only is the ratio of the X-force weapon system attrition rates against the Y-force constant, but also the ratio of the Y-force effectiveness against each of the two X-force types. Furthermore, all four attrition-rate coefficients have the same time dependence except for constant factors. The solution is shown in Table IV. In this special case it is seen that the structure of the optimal allocation policies when the attrition-rate coefficients are variable is essentially identical to that in which they are constant. Only the time scale has been transformed (in [22] this type of observation was first made by the author (see also [25])).

The development of the above results is omitted due to considerations of the length of the paper at hand. Their development and that of further such results are to be found in [28]. Moreover, it has been

important to include the above results here, since the purpose of the paper at hand is to contrast the solutions for a sequence of related problems.

Finally, it should be noted that when $h(t) = 1$ the results of Table IV reduce to those for Problem 1 given in Table II. To see this one sets $h(t) = 1$ and makes the following identifications: k_{a_1} in Problem 4 is replaced by a_1 in Problem 1, k_{a_2} by a_2, k_{b_1} by b_1, and k_{b_2} by b_2.

3.5. Fire Distribution for Targets Undergoing "Linear-Law" Attrition

So far in the state equations describing combat the attrition rate of each X-force target type has been proportional to only the number of Y-force firers. Considering Equations [7], [9], and [30] which give rise to the classical Lanchester square law, this may be referred to (somewhat imprecisely) as a "square-law" attrition process of target types.* H. Weiss [30] has given a thorough discussion of the conditions which lead to such an attrition process. These conditions include that "each unit is informed about the location of the remaining opposing units so that when a target is destroyed, fire may be immediately shifted to a new target." It is thus noted that the control theory models which we have considered so far have implicitly assumed perfect information in the above sense.

Another model for target type attrition is one in which the attrition rate (of each X-force target type) is proportional to the product of the numbers of targets and firers. This may be referred to (again somewhat imprecisely) as a "linear-law" attrition process of target types. Such an attrition process can arise under two different general circumstances: (1) fire is uniformly distributed over a constant target area ("area fire"), or (2) the mean time of target acquisition is much larger than target destruction time and is inversely proportional to target density. Again, quoting Weiss [30], it is assumed that units are informed about the general areas in which opposing units are located, but are not informed about the consequences of their own fire. Brackney [7] has shown that "aimed fire" may lead to a linear-law attrition process of targets when target acquisition times are considered and are as postulated above.

Thus, Problem 5 is a battle in which the attrition of each X-force target type is a linear-law process and is stated in mathematical terms below.

(Problem 5) $\underset{\phi(t)}{\text{maximize}}\{\text{ry}(T) - px_1(T) - qx_2(T)\}$ with T_1 specified,

$$\text{subject to: } \frac{dx_1}{dt} = -\phi a_1 x_1 y,$$

$$\frac{dx_2}{dt} = -(1-\phi) a_2 x_2 y,$$

$$\frac{dy}{dt} = -b_1 x_1 - b_2 x_2,$$

$$x_1, x_2, y \geq 0, \qquad 0 \leq \phi \leq 1, \qquad \text{and } T \leq T_1.$$

The analysis details upon which the summary given below is based are given in a companion paper [29] (see also pp. 91–105 of [21]) due to their rather lengthy nature. Moreover, it is important that these results be given here so that one can see the effect of the combat attrition model on the structure of the optimal allocation policy by contrasting, for example, the solution to Problem 1 with that for Problem 5.

*The reader should keep in mind that the Y-forces are faced with the problem of determining the optimal distribution of fire over X-force target types.

3.5.1. Description of Optimal Fire Distribution Policy

There is a fundamental difference between the solution to Problem 5 and those considered previously: the optimal allocation, ϕ^*, may be other than 0 or 1. In contrast to those for Problems 1 through 4, the optimal allocation policy does not have to be an extreme point of the control variable space at all times: one may have a singular solution [15] for which the necessary condition of maximizing the Hamiltonian (with respect to the control variable) does not provide a well-defined expression for the extremal control. That part of an optimal trajectory on which the maximum principle does not determine the control is called a singular subarc, and the term "singular solution" will be used to refer to any optimal trajectory which contains one or more singular subarcs.

Singular solutions usually occur when the Hamiltonian (denoted as H) is a linear function of the control variable(s). According to the notational conventions adopted in this paper, when this happens, then if $\frac{\partial H}{\partial \phi} = 0$ for a finite interval of time, the maximum principle does not determine the control. Observe that when $\frac{\partial H}{\partial \phi} = 0$ and H is a linear function of ϕ, *all* feasible values of ϕ are optimal. All problems, however, for which the Hamiltonian is a linear function of the control variables do not have singular subarcs in their solution. In particular, the reader should note that it has been shown above that it is impossible to have a singular solution to Problem 1 through Problem 4. This was done, for example, for Problem 4 by showing that it is impossible for $\frac{d}{dt}\left(\frac{\partial H}{\partial \phi}\right) = 0$ for a finite interval of time when $\frac{\partial H}{\partial \phi} = 0$. Moreover, there is a special second order necessary condition of local optimality (generalized Legendre-Clebsch condition) [16] which must be satisfied in order that a singular subarc can yield a maximum return. This is satisfied for the problem at hand [29].

The optimal battle trajectories are constructed by working backwards from all possible end points of this idealized battle [29]. Consideration is given to both the optimal control at the end of battle and also how the variables upon which it depends vary over time. Based upon such considerations, there are three cases to be considered:

$$\text{Case } (a) \qquad \frac{p}{q} = \frac{b_1}{b_2},$$

$$\text{Case } (b) \qquad \frac{p}{q} > \frac{b_1}{b_2},$$

$$\text{Case } (c) \qquad \frac{p}{q} < \frac{b_1}{b_2}.$$

Consider Case (a) first. The solution for this case is shown in Figure 2. Even though explicit expressions have not been obtained for certain model parameters, the dependence of the optimal control upon these quantities can still be qualitatively discussed. The optimal control depends on the state variables x_1 and x_2 (and also the attrition coefficients) in each "decision region." Above the line $a_1 b_1 x_1 = a_2 b_2 x_2$, denoted as L, the optimal control $\phi^* = 0$ is used until this line is encountered. When L is reached the singular control $\phi^* = \frac{a_2}{a_1 + a_2}$, which keeps the trajectory on L, is used until the end of the battle at $t = T$. That portion of an optimal trajectory which lies on L (for a finite interval of time) is a singular subarc. The

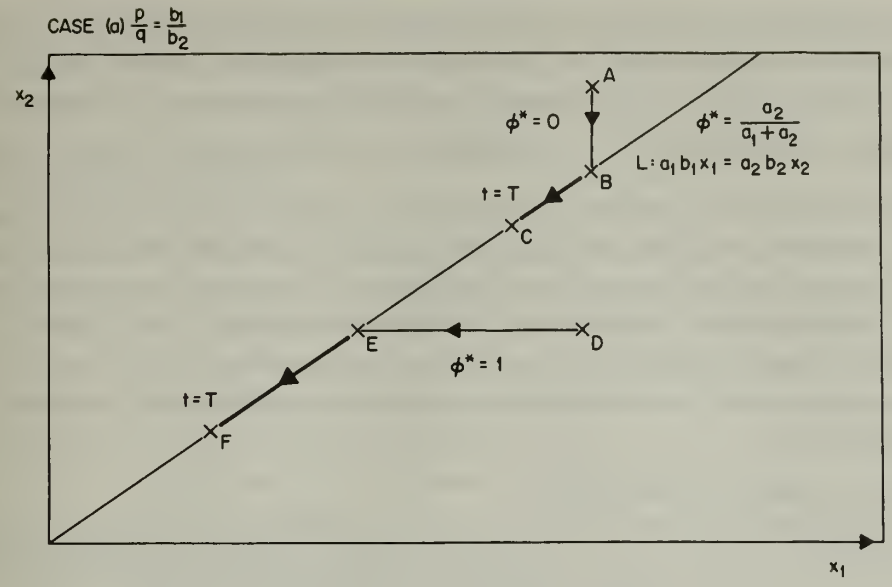


FIGURE 2. *Optimal allocation for linear-law attrition (Case a)*

time history of the optimal control is traced for two particular initial force ratios denoted as points A and D in Figure 2. At point D, $\frac{x_1^0}{x_2^0} > \frac{a_2 b_2}{a_1 b_1}$ and $\phi^* = 1$ is used until the line L is encountered at point E.

The solution for Case (b) is shown diagrammatically in Figure 3. It is similar to the preceding case except that another line, L' with equation $a_1 p x_1 = a_2 q x_2$, plays a role in the solution in addition to the singular "surface" denoted as L. This line L' appears above, on, or below the line L (with equation $a_1 b_1 x_1 = a_2 b_2 x_2$) depending upon whether $\frac{\dot{p}}{q}$ is greater than, equal to, or less than $\frac{b_1}{b_2}$.

The significance of the line L' and its relationship to the line L is as follows. The battle is divided into two time phases: Phase I for $0 \leq t \leq t_I = T - \tau_I$ and Phase II for $T - \tau_I = t_I \leq t \leq T$. During Phase I

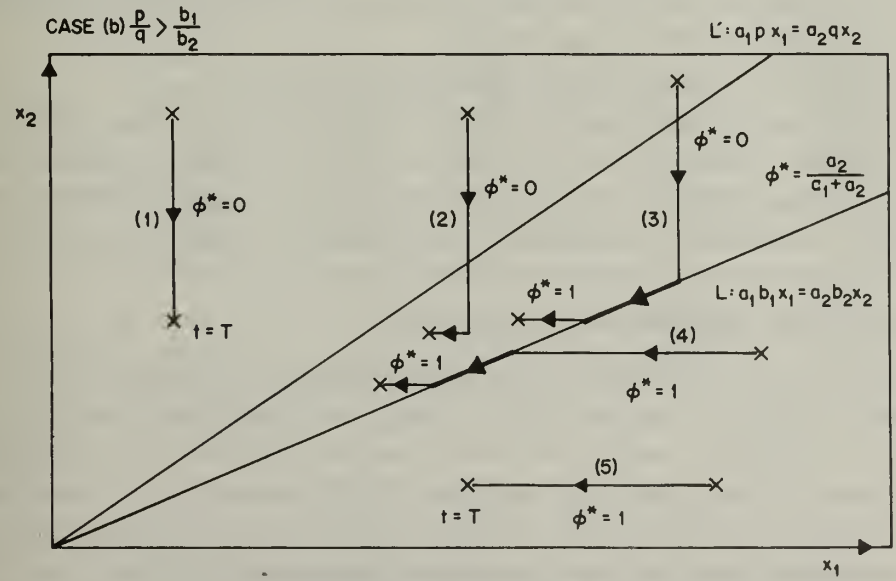


FIGURE 3. *Optimal allocation for linear-law attrition (Case b)*

the optimal target engagement policy at a point in time is determined by the location of the point on the battle trajectory with respect to the line L, which is also the singular "surface." Above L, $\phi^*(t)=0$; while below L, $\phi^*(t)=1$. When a battle trajectory reaches L, it remains on the singular surface through use of the singular control $\phi^*=\frac{a_2}{a_1+a_2}$. During Phase II the optimal target engagement policy is to use $\phi^*(t)=1$ below L' and $\phi^*(t)=0$ above L'. The proof of the above statements is given in [29].

Thus for any optimal trajectory which lies below L' (such as those denoted in Figure 3 as (2), (3), and (4)) the optimal control is $\phi^*=1$ during Phase II. Moreover, the time $t_I=T-\tau_I$ appears in Figure 3, for example, as when the optimal control for (3) switches from $\phi^*=a_2/(a_1+a_2)$ to $\phi^*=1$. Any optimal trajectory which lies entirely above L', such as (1), has a corresponding optimal control of $\phi^*(t)=0$ for $0\leq t\leq T$, whereas a similar remark holds for any one that lies entirely below L, such as (5). Case (c) is symmetric to Case (b).

3.5.2. Discussion of Structure of Optimal Policy

As noted above (see sections 3.1.2. and 3.3.1.), the structure of the optimal allocation policies in these tactical allocation problems is dependent upon how the Y-force values the surviving X-force types relative to their kill rate against the Y-force. Case (a): $\frac{p}{q}=\frac{b_1}{b_2}$ above is when Y assigns utility to surviving X-force types in exact proportion to their destructive capability against Y. In this case, the optimal target selection tactic depends only upon the state of the system as is seen with reference to Figure 2. The optimal tactic is to use $\phi^*(t)=0$ above the line L with equation $a_1b_1x_1=a_2b_2x_2$. The line L also represents an "equilibrium" trajectory which the system follows whenever this line is reached. Case (*b*): $\frac{p}{q}>\frac{b_1}{b_2}$ is when Y assigns a greater value to surviving X_1's than in proportion to their kill rate against Y relative to that of X_2. Again, the optimal tactic depends upon the state of the system, only this dependence itself depends upon the "time phase" of the fixed length battle.

Based on the above examination of Problem 5, it is seen that the structure of the optimal allocation policies for targets which undergo a linear-law attrition process has the following characteristics:

(1) fire may be divided between target types,

(2) the allocation is (directly) dependent upon the force levels.

These characteristics should be contrasted with those previously observed when target types undergo a square-law attrition process (see sections 3.1.2. and 3.3.3.). When there is a linear-law attrition process of target types, the optimal allocation policy may be other than 0 or 1. Also, the allocation depends upon the force levels of target types. An explanation for this structure of optimal allocation policies in terms of the nature of the attrition process has been given in section 3.1.2.

4. EXTENSIONS TO DIFFERENTIAL GAMES

Even though it is certainly true that combat is an environment of conflicting interests in which the potential actions of both friendly and enemy forces must be considered, there is much to be learned from one-sided dynamic optimization models. The author views these simplified idealizations presented here as "building blocks" for more sophisticated scenarios. It is felt, moreover, that an understanding of the structure of optimal tactics for these initial models is essential before one continues his examination of a sequence of models of greater and greater complexity. Hence, it seems appropriate to review the intimate connection between optimal control theory and differential games.

It has been stated that optimal control problems may be viewed as one-sided differential games for which the roles of all but one of the competing players have been suppressed [2]. Conversely, differential games may be viewed as two-sided optimal control problems [12]. A concise discussion of the interrelationships between these two subjects is contained in Y. C. Ho's [11] excellent review of Isaacs' book [13] (see also chapter 9 of [8]).

It should be recalled (see [24]) that the existing theory of (zero-sum deterministic) differential games is only applicable to problems for which the criterion functional has a saddle point in pure strategies. However, it may be shown (considering the results of A. Friedman (see chapters 5 and 6 of [10])) that the structure of two-sided fire distribution problems in the Lanchester theory of combat as formulated by Isbell and Marlow [14] and Weiss (see pp. 94–95 of [30]) guarantees the existence of pure strategy solutions. This need not be true for other dynamical structures. For example, when defensive capabilities were considered in the attrition process in a tactical air war game extensively studied at RAND, the resulting model did not possess a solution in pure strategies [1], [3], [4].

The author has therefore, used these optimal control problems to study many aspects of such corresponding differential games (two-sided variational problems): the effect of different boundary conditions, devising solution procedures, study of singular behavior, differences in the structure of optimal allocation policies for various model forms. Most solution aspects of the one-sided problem are present in the two-sided one. (There are exceptions, however (see [26]).) In solving [26] the supporting weapon system game of H. K. Weiss [31], the author made use of his knowledge of a related optimal control problem [24].

5. IMPLICATIONS OF MODELS

It seems appropriate to briefly discuss the general implications of the models examined in this paper to the following areas:

(1) optimal tactical allocation,

(2) intelligence,

(3) command and control systems,

(4) human decision making.

The discussion of these areas is not mutually exclusive.

Of interest to the military tactician is whether optimal fire distribution rules evolve dynamically during the course of battle. Are target priorities static or do they evolve dynamically with the course of battle? With respect to optimal control models, this may be mathematically stated as whether there are transition (switching) surfaces in the solution. It has been seen in the idealized and simplified models studied here that target priorities do change. This is related to the evolution of marginal return of target destruction (value of dual variable). It has been seen that this evolution depends on the goals of the combatants (utility assigned to surviving force types at the end of the battle) and also the conditions which terminate the battle. In the terminal control problem studied here, a shift in target priorities is present only in a losing case, whereas in a fixed duration battle such a switch is sometimes independent of winning or losing and then depends only on weapon system capabilities and the prescribed duration of battle.

Schreiber [20] has proposed an idealized and simple, but yet illuminating, way of quantitatively showing the value of intelligence and command control capabilities. He introduces the concept of "command efficiency," which is measured by the fraction of the enemy's destroyed units from which fire has been redirected. The effect of poor intelligence and poor capabilities for redirecting fire from

destroyed targets is to produce "overkill." Schreiber's equations for combat involved this fraction called "command efficiency," and they reduce to Lanchester-type equations for area fire when the fraction is 0 and aimed fire for a value of 1. It has been seen that the optimal tactics are quite different for these two cases. When intelligence and command control systems are very efficient, the optimal tactic is seen always to be concentration of fire on a specific target type. When capability for redirection of fire from destroyed targets is poor (either through damage assessment or constraints on new target acquisition), the optimal tactic may be to allocate fire in a proportional fashion over target types in a way that holds the ratios of target density in each target area to be constant. Thus, these models indicate that the optimal tactics of fire distribution vary with command and control capabilities.

These models also show the importance of intelligence in devising the "best" tactics in combat. Intelligence on enemy weapon system capabilities (kill rates including target acquisition rates) and potential length of engagement play a central part. It has also been seen that for fights-to-the-finish and linear-law attrition cases intelligence on enemy force levels is also required. For artillery fire support missions against various troop concentrations, knowledge of troop densities is essential in the assignment of target priorities. Particularly dense concentrations where the initial kill potential is high are seen to be cases where the optimal tactic is to concentrate fire on one target for awhile.

These models may be interpreted to show the value of human judgment in combat. They indicate, as does common sense and experience, that in battle a commander must use his judgment to ascertain to what end can the course of battle be steered so that he may devise his strategy accordingly. The demonstrated sensitivity of these models to many factors shows the importance of human assessment of a situation and the importance of good judgment in assigning utility to forces surviving the battle at hand.

6. SUMMARY

The results of this paper may be summarized as follows:

(1) a sequence of one-sided models has been presented which shows that the tactics of fire distribution are sensitive to force levels, target acquisition process, the type of attrition process, and the termination conditions of combat,

(2) tactics for target selection are heavily dependent upon "command efficiency,"

(3) concentration of fire always on one target type among many occurs as an optimal tactic only when target acquisition is not subject to diminishing returns,

(4) target priorities do not change over time when one assigns a worth to surviving target types in direct proportion to their kill rate against you.

7. ACKNOWLEDGMENT

The author would like to thank the referee for his suggestions for improving this paper.

REFERENCES

[1] L. Berkovitz, "A Differential Game with no Pure Strategy Solution," in *Advances in Game Theory*, M. Dresher, L. Shapely, and A. Tucker (Eds.), (Princeton University Press, Princeton, 1964), pp. 175–194.

[2] L. Berkovitz, "Necessary Conditions for Optimal Strategies in a Class of Differential Games and Control Problems," SIAM J. Control *5*, 1–24 (1967).

[3] L. Berkovitz and M. Dresher, "A Game Theory Analysis of Tactical Air War," Opns. Res. *7*, 599–620 (1959).

[4] L. Berkovitz and M. Dresher, "Allocation of Two Types of Aircraft in Tactical Air War: A Game Theoretic Analysis," Opns. Res. *8*, 694–706 (1960).

[5] S. Bonder, "The Lanchester Attrition-Rate Coefficient," Opns. Res. *15*, 221–232 (1967).

[6] S. Bonder, "The Mean Lanchester Attrition Rate," Opns. Res. *18*, 179–181 (1970).

[7] H. Brackney, "The Dynamics of Military Combat," Opns. Res. *7*, 30–44 (1959).

[8] A. Bryson and Y. C. Ho, *Applied Optimal Control* (Blaisdell Publishing Co., Waltham, Massachusetts, 1969).

[9] L. Dolansky, "Present State of the Lanchester Theory of Combat," Opns. Res. *12*, 344–358 (1964).

[10] A. Friedman, *Differential Games* (Wiley-Interscience, New York, 1971).

[11] Y. C. Ho, "Review of the Book *Differential Games* by R. Isaacs," IEEE Trans. on Automatic Control, Vol. AC–10, 501–503 (1965).

[12] Y. C. Ho, A. Bryson, and S. Baron, "Differential Games and Optimal Pursuit-Evasion Strategies," IEEE Trans. on Automatic Control, Vol. AC–10, 385–389 (1965).

[13] R. Isaacs, *Differential Games* (John Wiley and Sons, Inc., New York, 1965).

[14] J. Isbell and W. Marlow, "Methods of Mathematical Tactics," *Logistics Papers*, No. 14, The George Washington University Logistics Research Project (Sept. 1956).

[15] C. Johnson and J. Gibson, "Singular Solutions in Problems of Optimal Control," IEEE Trans. on Automatic Control, Vol. AC–8, 4–15 (1963).

[16] H. Kelley, R. Kopp, and H. Moyer, "Singular Extremals," in *Topics in Optimization*, G. Leitman (Ed.) (Academic Press, New York, 1967), pp. 63–101.

[17] B. Koopman, "The Optimum Distribution of Effort," Opns. Res. 1, 52–63 (1953).

[18] J. McIntyre and B. Paiewonsky, "On Optimal Control with Bounded State Variables," in *Advances in Control Systems, Vol. 5*; C. Leondes (Ed.) (Academic Press, New York, 1967), pp. 389–419.

[19] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko, *The Mathematical Theory of Optimal Processes* (Interscience, New York, 1962).

[20] T. Schreiber, "Note on the Combat Value of Intelligence and Command Control Systems," Opns. Res. *12*, 507–510 (1964).

[21] J. Taylor, "Application of Differential Games to Problems of Military Conflict: Tactical Allocation Problems—Part I," Naval Postgraduate School Technical Report No. NPS 55TW0062A, Monterey, California (June 1970).

[22] J. Taylor, "A Note on the Solution to Lanchester-Type Equations with Variable Coefficients," Opns. Res. *19*, 709–712 (1971).

[23] J. Taylor, "Comments on a Multiplier Condition for Problems with State Variable Inequality Constraints," IEEE Trans. on Automatic Control, Vol. AC–17, 743–744 (1972).

[24] J. Taylor, "On the Isbell and Marlow Fire Programming Problem," Nav. Res. Log. Quart. *19*, 539–556 (1972).

[25] J. Taylor, "Comments on 'A Note on the Solution to Lanchester-Type Equations with Variable Coefficients,'" Opns. Res. *20*, 1194–1195 (1972).

[26] J. Taylor, "Comments on Some Differential Games of Tactical Interest," Opns. Res., to appear.

[27] J. Taylor, "On the Solution to Lanchester-Type Equations of Modern Warfare with Variable Coefficients," Opns. Res., to appear.

[28] J. Taylor, "Target Selection in Lanchester Combat: Heterogeneous Forces and Time-Dependent Attrition-Rate Coefficients," to appear in Nav. Res. Log. Quart.

[29] J. Taylor, "Target Selection in Lanchester Combat: Linear-Law Attrition Process," to appear in Nav. Res. Log. Quart.

[30] H. Weiss, "Lanchester-Type Models of Warfare," *Proc. First International Conf. Operational Res.*, Oxford (1957).

[31] H. Weiss, "Some Differential Games of Tactical Interest and the Value of a Supporting Weapon System," Opns. Res. 7, 180–196 (1959).

EXPERIMENTS IN COMMUNICATION NETWORKS*

G. M. Cady
System Development Corp.

B. P. Lientz
University of California, Los Angeles

and

N. E. Willmorth
System Development Corp

1. INTRODUCTION

Consider a computation oriented communications network where nodes represent computing centers and links are communication ties, such as hard lines, microwave, or satellite. The parameters in such a network include the properties of hardware and software computation, communication links, and message traffic across the network. Such a system represents several government existing and proposed networks.

The purpose of such a network is several fold. First, reliability can be increased by providing rapidly accessed alternative centers. A second purpose lies in the cost effectiveness gained through more efficient utilization of computer resources. There have been several references on the analysis of computer networks and computation (see, for example, Kleinrock [9]).

In considering such networks, several important questions arise. One involves the degree of centralization of computing power that results in the most cost beneficial designs. For example, distributed centers reduce vulnerability, but increase some maintenance expenses. Because of the number of parameters and their interrelationships, there may not be one general answer under all conditions. In this case, the goal of experimentation is first to find those parameters which have the most influence on the configuration, and then secondly to find the most effective configuration for ranges of those parameters. The experiments for analysis of centralization are described in section 2. The conclusion of these set of experiments is that semi-centralized configurations (large computers in several cities) are most cost-effective under a variety of conditions.

The relationship between job size and memory size is explored. It should be emphasized that the conclusions reached here are based on the ranges of parameters involved. For precise evaluation, the specific network being considered would have to be used.

Another question involves the method of designing the communications link structure. Some results are summarized in Frank and Frisch [7]. Many approaches however, have been graph theory oriented rather than dependent upon the anticipated or actual message traffic. Some related graph theoretic problems in communication networks appear in Lientz [10] and in Alter and Lientz [3], [4] for radar and tracking situations.

*This work was partially supported under an Office of Naval Research Contract N00014-67-A-0269-0027

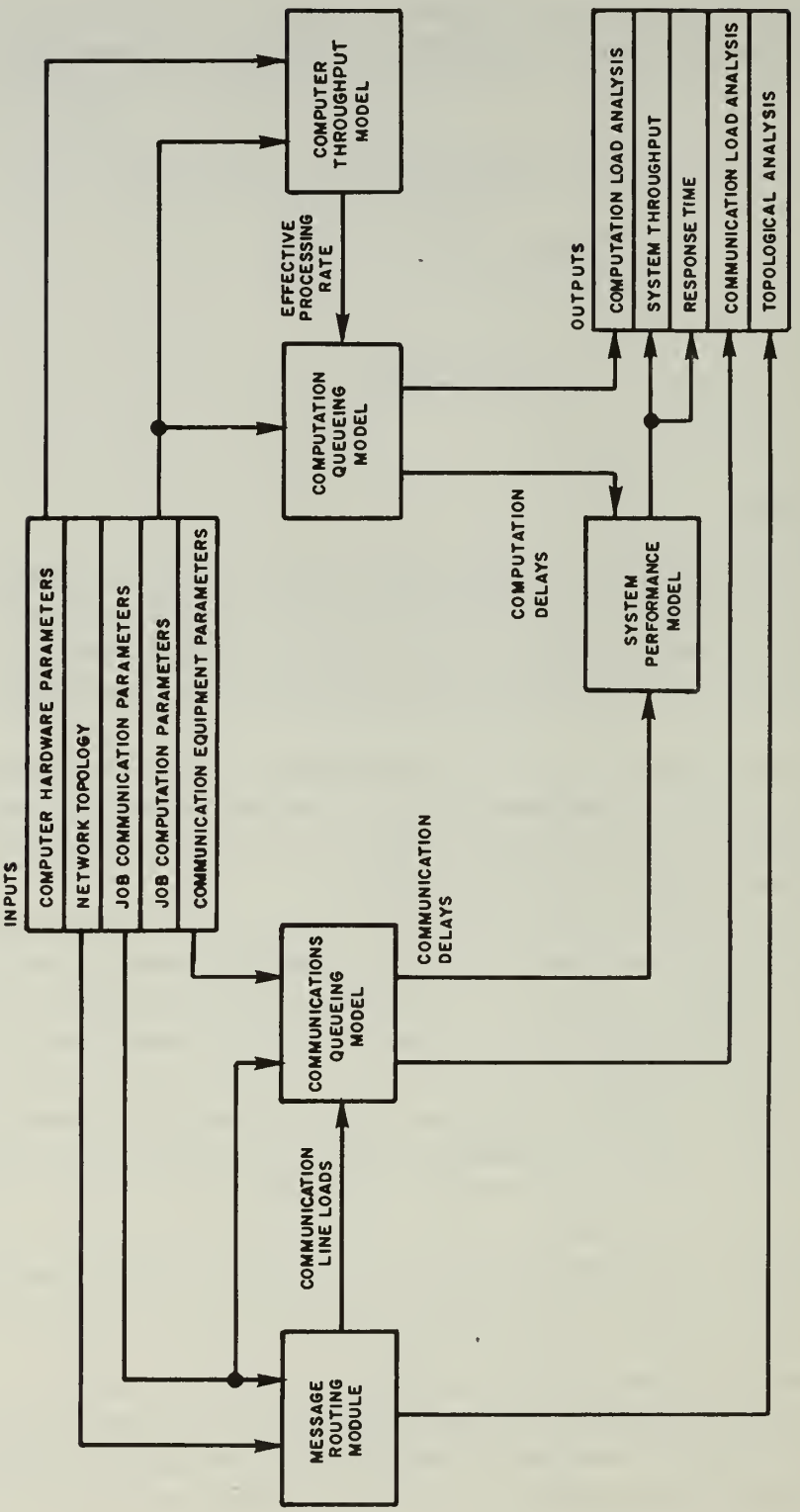


FIGURE 1. Functional flow of information in analysis model.

Other approaches have been heuristic in that a configuration has been specified because of a particular property. A central property is that of articulation level. A network has articulation level K if exactly K (and no less) nodes have to be destroyed so that at least two surviving nodes cannot communicate.

Articulation level 2 is sometimes specified so as to provide at least one communication path in case of deterioration or failure. Section 3 presents a set of experiments aimed at considering several methods of link construction based on cost effectiveness and dependent upon the message traffic and reveals that this is more cost-effective than standard configurations or uniform traffic loading.

The experiments were conducted using an analytic FORTRAN IV program designed for computation oriented networks. Most analysis tools for these networks have been either computer simulators such as SKERT (analytic) and ECSS (discrete) or communications oriented models. A combined computer and communication analytical analysis program was constructed for performing analysis tasks. This program is described in Cady [5]. The functional layout of the program is shown in Figure 1.

The assumptions of the model are given in [5] and discussed in section 2. Basic assumptions involve those based on queueing theory. Some were developed in Kleinrock [8]. The program imbeds a near optimal fixed link routing logic.

Validation of the methodology was carried out in several phases. The communications analysis methodology was based on the work of Kleinrock [8], and using a discrete simulation program in ECSS.

The hardware computation equations were validated using some of the extensive IBM Sort/merge statistics with 20 configurations. Best fit was obtained with moderate to large core (at least 256 K bytes).

The third part of the validation concerned the software aspects of computation including the effect of record size, I/O—cpu balance and overlap. Validation was performed using a small JOVIAL program which carried out a specific number of I/O and cpu operations and permitted internal timing (JOVIAL allows such timing).

2. CENTRALIZATION OF COMPUTATION

In order to evaluate the degree of concentration of computation, several initial experiments were carried out to determine the parameters for variation. The assumptions used to perform the experiments were as follows:

(a) Remote and local jobs were both considered. A remote job consists of a message transmission, computation, and a return message. A local job is entirely computation. Messages were assumed to have single sources and destinations.

(b) All messages between two nodes followed the path with the minimum number of links between the nodes (fixed minimum path routing) with ties for the minimum length path being resolved by assignment to the least loaded path.

(c) Message and job arrival distributions were assumed to be negative exponential.

(d) Interarrival times were independent of message lengths and job sizes.

(e) Nodes behaved independently of each other. This assumption implies infinite-capacity message buffers.

(f) Node switching delays were fixed.

(g) Nodes were assumed to have an infinite traffic capacity.

(h) Multiprogramming and multiprocessing were not specifically accounted for in the model.

(i) Message transmission was assumed to be error-free. Retransmission is not explicitly taken into account.

Some comments on these assumptions are in order. The fixed minimum path routing has been shown to be close to optimal and requires much less computation than is required for calculations of the optimal case (Frank [6]), although arrival times are more closely approximated by gamma distributions. However, in many cases (especially for summed gamma distributions), the effective differences between the gamma and exponential distributions have been demonstrated to be small. Kleinrock [8] has shown that this occurs when all users on a large network are considered simultaneously. The assumption of infinite-capacity message buffers has been shown to be valid when the network is operating at less than 80-percent capacity. In a network in an unsaturated state with minimum time delay, the limitations on node capacity are minor (Kleinrock [8]).

The framework for the experiments was a 40-node network with centers located in the cities listed in Table 1 (the numbers in parentheses indicate the number of centers in the given city).

TABLE 1. *Cities for Experiment Set I*

Index of City	City	Index of City	City
1	Seattle	14	Denver
2	Buffalo	15	Cincinnati
3	Boston (2)	16	San Francisco
4	Portland	17	Kansas City
5	Milwaukee	18	St. Louis
6	Minneapolis	19	Los Angeles (3)
7	Detroit (2)	20	Phoenix
8	New York (6)	21	Atlanta
9	Chicago (3)	22	San Diego
10	Pittsburgh	23	Dallas
11	Philadelphia (2)	24	New Orleans
12	Cleveland	25	Houston
13	Washington, D.C. (3)	26	Miami

These cities were selected in part from geographic distribution and in part from population and density statistics. Each center was given one or more specified computers which were tied into the network by a standard modem device.

The general network topology appears as in Figure 2. Distances were computed by the model using latitude and longitude data (computer centers in the same city were assumed to be 3 miles apart). The network's topology was constructed iteratively, where at each step the most highly loaded group of links was found. Then a link was added to reduce this load. A lightly loaded link elsewhere in the network was deleted. This proceeded in an enumerative way until no further improvement was possible.

The capacity of each communication line was set at 50kb. This channel capacity bears a desirable ratio of communication to computation capacity and reflects the optimum ratio of communication to computation costs found in past experiments of other researchers for second-generation computing experiment. To be representative of third-generation capabilities, the configuration of the network was based on the assumption that the articulation of the network was two—that is, at least two links must be broken to break communication between two centers. The monthly communication cost was based on

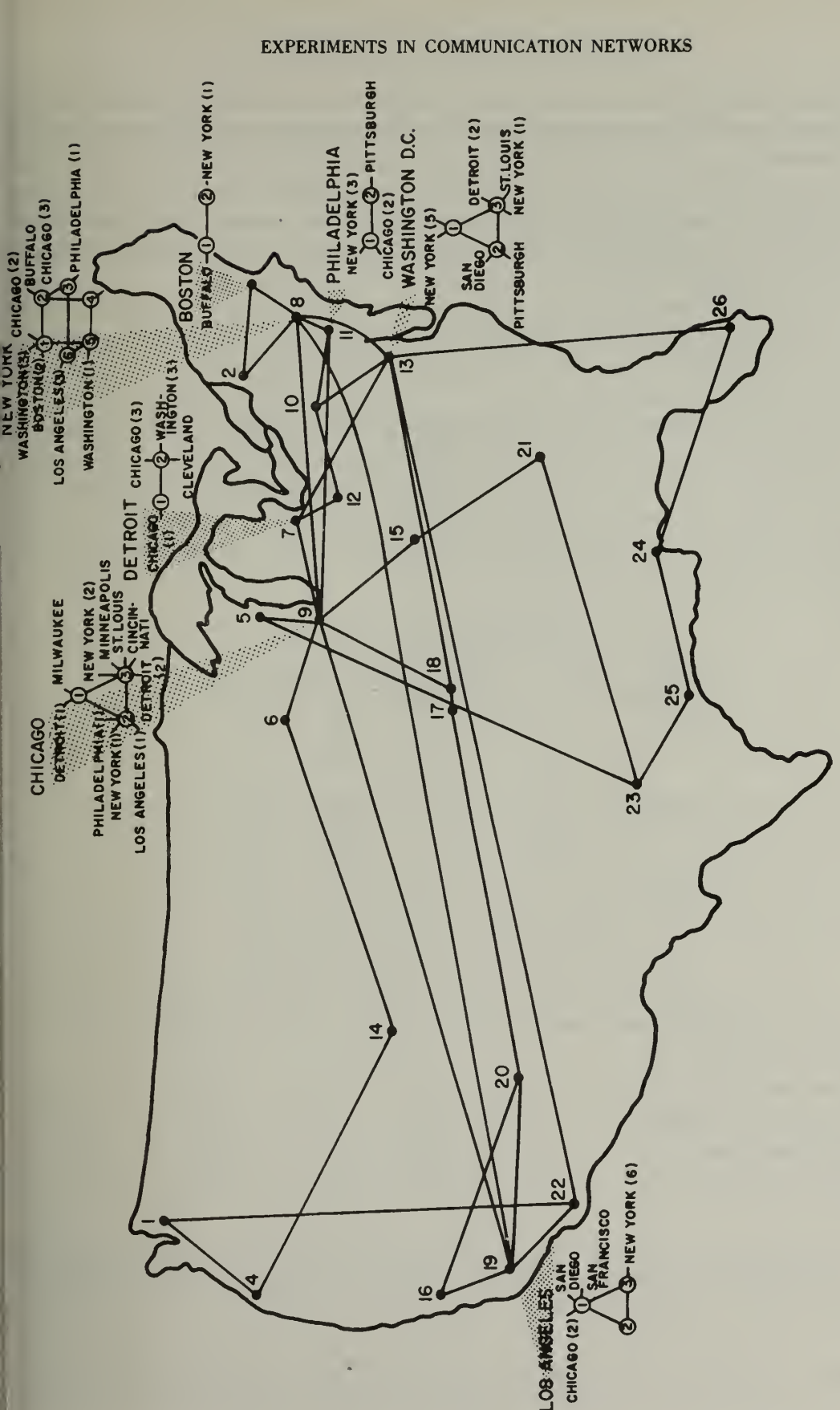


FIGURE 2. Network configuration for centralization.

standard available rates for 50kb line size, given by $15.00 for each of the first 250 miles, $10.50 for each of the next 250 miles, and $7.50 for each mile beyond 500 miles. A minimum cost of $250 per month per line was also assumed.

To obtain a representative set of costs and performance, a single hardware manufacturer and generation were selected. This was the IBM 360 line of machines. This line was chosen because of the broad range of compatibly operating equipment and a consistency of costs not yet present in the latest product lines. The machines are listed in Table 2.

The cost and configuration information on the computers and peripherals used—360/20, 360/85, and 360/195—appears in Tables 2 and 3. The 360/85 was assumed to have 2,314 disc units, while the 360/195 was given 3,330 units. The cost information reflects the costs of peripherals and the mainframe computing unit on a monthly rate.

Three basic job mixtures were chosen to reflect a variety of network situations. The main parameter here is the percentage of time the average job spends in CPU. This ranges from 90 percent for scientific to 10 percent for commerical. To show the sensitivity of computer throughput to core memory size, and to investigate the relationship between job type characteristics and memory, three levels of immediate memory were investigated for each computer. Details on monthly rental price and performance were obtained from Keydata [1] and Auerbach [2].

TABLE 2. *Experiment Set I—Computer costs*

Model	Memory (1000 bytes)	Monthly Cost (000)
360/85	500	$85
	1000	150
	2000	200
360/195	1000	215
	2000	258
	4000	300
360/20	16	6

TABLE 3. *Table of Computer Characteristics**

Characteristics	360/85	360/195
Job processing rate (instruc. microsec.)	6.25	21
Memory size (thousands of bytes)	500, 1,000, 2,000	1,000, 2,000, 4,000
Word size (bits)	32	32
Disk transfer rate (bytes/microsec.)	312	806
Average disk access time (millisec.)	87.5	38.5
Disk cylinder size (bytes)	146,000	247,600
Average I/O record size (bytes)	7,224	7,224

The preceding information provides the general topology and cost framework. Remaining to be specified are the job and message characteristics, as well as the specific combinations of hardware for the experiments.

*The 360/20 is included only in communication costs, and its computational characteristics are not included in the experiment.

Three different computation configurations were assumed for the experiments—distributed, semi-centralized, and centralized computing power. The detailed assignments of computers are given in Table 4. (The number in parentheses refers to the location within the city.) In all cases, the 360/20 computers act as concentrators and message processors. The 360/195 computers are split pairwise in a dual processing mode. In each case, the total raw throughput capacity of the configurations was roughly equivalent (approximately 8 billion modified bits per second). For the 360/195, this takes into account a 15-percent loss, resulting from executive software overhead in coordinating the dual processors.

TABLE 4. *Experiment Set I — Computation Configuration*

Configurations	Computer Assignment
Distributed	360/85 at all sites
Semi-centralized	360/195—2 at each of Boston (2) New York (5) Chicago (3) Washington, D.C. (3) St. Louis Los Angeles (1) 360/20—other centers
Centralized	360/195—2 at each of the 6 New York centers 360/20—other centers

The mixture of job types is similar to that experienced with a general-purpose computer utility (on-line, interactive operations meshed with remote-job-entry, non-interactive batch processing). These job types were chosen to emphasize extremes of job mixes and to provide information about the relative merits of centralized and distributed processing power for various job types. Further, the job configurations are directly related to the throughput efficiency of the different computer configurations also being evaluated in these experiments.

The experiments assumed that jobs consist of messages decomposed into packets. A packet is the basic unit of bits in the communications part of the experiments. The packet size was set at 2,000 bits.

Several other parameters that had to be determined were job size and the job-arrival matrix. The job arrival matrix has as its (i, j)th entry the number of jobs sent from i to j. The job size was allowed to be variably set at one of three values—5, 10, and 20 megabits. It was recognized that some combinations would be unrealistic. The job-arrival matrix was first set for the distributed case and then centralized as the computer configuration centralized. To derive the number of jobs arising from a given city, the proportion of city population to the total population in all cities was multiplied by the total permissible jobs. The creation of a traffic matrix was based upon the relative distance of the computing centers from the source cities. In the completely centralized case, the job load was distributed equally among the centralized computers. Jobs arising locally around a centralized computer were all assigned to that computer. In the semi-distributed case, where the computers were dispersed to locations about the nation, the traffic was distributed to the processing centers as the

square root of the distance from the source city, normalized to the sum of the square root distances to obtain a proportion of traffic. For the completely distributed case, 50-percent of the jobs arising at a source city were assigned to the computer at that city. The remaining jobs were distributed among all other cities by the square-root distance formula used above. The selection of square root of distance was based on reducing loads between distant cities somewhat, but not to an excessive degree. Another assumption of the message traffic was the allowance for acknowledgment messages.

The assumption that jobs arise in proportion to population has been made in previous network analysis reports in connection with message traffic. Dependence on the populations of cities at both ends of the link can reflect the difference in computing power for major centers.

The absolute level of work was based initially upon an estimated 70-percent utilization rate of the raw processing power (i.e., total megabits modified per second by all computers) of the system. This total utilization level was adjusted during the experimental runs to reflect the reduced throughput resulting from the assumptions concerning job characteristics.

The above framework established a set of 81 distinct experiments in which three values of each of the following parameters were set: job size, configuration, job type, and core size. Total cost of the network configuration varied from $3.1 to $8.3 million per month.

Based on this set of experiments inference can be made concerning the relationship between system input parameters and network performance measures. The basic network performance measures are cost, response time, job throughput, and measures of cost-effectiveness.

Costs included the entire monthly costs associated with the computer and communication system hardware, including the computer peripherals and memory units, communication interface units (modems, switches, concentrators, etc.), initial costs of the central processing units, and direct channel costs.

Response time was defined as the mean time for both computation and communication processing, often involving the averaging of combinations of times from several computation nodes and communication links. However, initial generation and output distribution times were ignored; only network times were considered.

Throughput was defined as the number of jobs processed per day, modified by considerations of job and message size. Throughput is thus the workload of the system, rather than system capacity.

The principle measure of *cost effectiveness* was the quantity throughput per dollar per unit response time. That is, the total throughput (number of jobs times job size) was divided by the product of total monthly cost and the mean total response time.

In interpreting the results, the fact that several network configuration and job characteristic parameters were held constant needs to be kept in mind. Total computation and communication capacity was held roughly constant for all configurations (the capacity for computation was set at 250 million instructions per second, while the line capacities were fixed at 50kb for communication. The network topology, except for the distribution of computing capacity, was fixed (as described in the previous section). Although central memory capacity was varied, all other aspects of the computer facility configuration were held constant for a given computer. Hence, cost varied with the constellation of computing processors and memories used, but not with other (fixed) aspects of the computer or communication configuration.

The results of the 81 experimental runs are given in Table 5 in a nested arrangement. The code for concentration, given in column (1), is:

D—Distributed
S—Semi-centralized
C—Centralized

The amount of core memory associated with each main processor is given in megabytes in column (3). Columns (4), (6), and (8) give the experiment number, and columns (5), (7), and (9) the cost-effectiveness (CE) index for each of the 81 experimental runs. These sets correspond to the three job sizes, set one for jobs of 5 megabits (MB), set two for jobs of 10 MB, and set three for jobs of 20 MB. (Note that at the 20–MB job size, two experimental runs—numbers 61 and 79—exceeded the capicity of the system—mean and response time was very long—and a zero cost-effectiveness index is indicated.) Cost-effectiveness is defined here as the square root of the throughput divided by the product of (*a*) total network costs squared and (*b*) the mean total response time.

TABLE 5. *Results of Experiments*

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Concentration	Job type	Core (MB)	5 MB		10 MB		20 MB	
			Expt. No.	C.E. $\times 10^{-2}$	Expt. No.	C.E. $\times 10^{-2}$	Expt. No.	C.E. $\times 10^{-2}$
D	S	5	1	19896	28	20034	55	18826
D	S	1	2	13337	29	13743	56	13367
D	S	2	3	11014	30	11506	57	11400
D	M	5	4	08531	31	08730	58	04070
D	M	1	5	05565	32	06242	59	05581
D	M	2	6	04475	33	05259	60	05297
D	C	5	7	03806	34	04719	61	0
D	C	1	8	02292	35	03032	62	03200
D	C	2	9	01763	36	02382	63	02929
S	S	1	10	36235	37	41341	64	43268
S	S	2	11	32794	38	38493	65	41003
S	S	4	12	29631	39	35030	66	37785
S	M	1	13	11870	40	15259	67	15740
S	M	2	14	10421	41	13754	68	15799
S	M	4	15	09212	42	12293	69	14561
S	C	1	16	03769	43	05247	70	05188
S	C	2	17	03241	44	04540	71	04751
S	C	4	18	03594	45	03983	72	04191
C	S	1	19	34338	46	35514	73	44019
C	S	2	20	31125	47	36931	74	42027
C	S	4	21	27811	48	33491	75	38843
C	M	1	22	11070	49	14415	76	15553
C	M	2	23	09689	50	12909	77	16447
C	M	4	24	08546	51	11498	78	15301
C	C	1	25	02836	52	03978	79	0
C	C	2	26	02439	53	03429	80	06173
C	C	4	27	02134	54	03003	81	05501

Although some caution must be exercised in interpreting the results since only a few of the myriad possible variables and combinations of variables were manipulated, some conclusions seem clear. For instance, the productivity of a particular configuration is related in a direct and dramatic fashion

to the degree of CPU utilization that is achieved. The cost-effectiveness index falls drastically as the fraction of time in computation changes from 90 percent (scientific) to 50 percent (mixed) and 10 percent (commercial). While the inefficiency of I/O-bound jobs is generally accepted in the computation and communication field, that the interrelationship should be so severe was not entirely expected. The dependence upon job mix applies only to a batch processing mode. A referee has pointed out that in certain situations the dependence would be greatly weakened.

A similar instance is the growth of productivity with computing load (job size, in this case). The cost-effectiveness index continues to increase until the system or processor becomes saturated, after which it begins to fall off rapidly. In other contexts, the efficiency of channel utilization in terms of response time has been found to decline, depending upon a variety of circumstances, in the 70–90 percent channel-utilization points. While there are not enough data points in this study to make such fine distinctions, the general premise is supported. Since the relationship between load and response efficiency is well established, the experiments were designed not to reaffirm it, but to explore some of the interrelationships between job size and memory size and the resulting cost-effectiveness.

The relationship between job size (load) and memory size seems reasonably clear. The efficiency of core-storage additions increases as the load on the processor increases. That is, from our data, at a relatively light load (jobs of five MB) small core is more cost-effective; at relatively high loads (jobs of 20 MB), larger core becomes more cost-effective. The shift is a gradual one; however, even with loads large enough to swamp computers with relatively small core sizes, a moderate core is more cost-effective than a very large one. This is revealed in Figure 3, which graphs job size versus cost-effectiveness

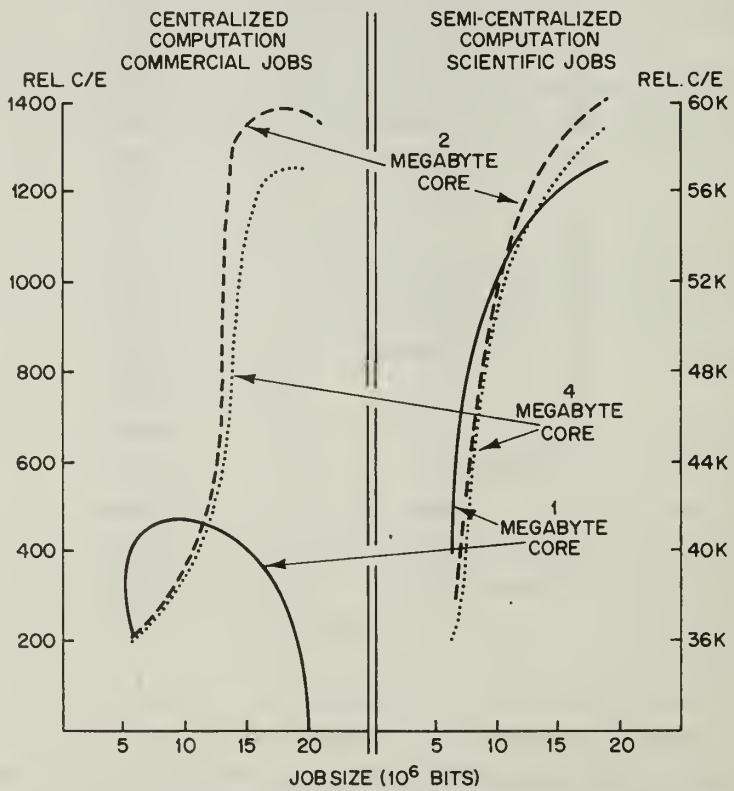


FIGURE 3. Memory size cost-effectiveness.

for various core sizes. It might be pointed out that system response time continues to improve with increasing core sizes. It is the disproportionate cost of extra core (in comparison to the increase in cost-effectiveness) that inhibits the strength of the relationship.

Of major interest in this investigation is the relationship between distributed and centralized computing. The difference in relative concentration of computing power between the completely centralized and the semi-distributed cases lies in the distribution of large computers by location around the United States. That is, the centralized case does not use one super-powerful computer, but a concentration of 12 very large ones, a situation that is duplicated with different locations in the semi-distributed case. The data clearly support the hypothesis of economy of scale; large computers are much more cost-effective than smaller computers (although all computers considered in this study were quite large) and especially so as the load becomes high. That is, with high load into a constant-capacity net, the smaller computers suffered in comparison with the larger. Similar results have been found in the communication channels, where, for constant capacity, one large line has been found more efficient than a bundle of smaller ones insofar as throughput is concerned. For response time, there is some evidence that multiple channels or processors yield better results in speed, but not necessarily in terms of cost-effectiveness.

From the experimental results, the semi-distributed configuration appears more cost-effective than the completely centralized case, partially as a result of shorter, and hence more quickly responding, communication lines. No attempt was made to adjust communication network capacity or cost to accommodate differences in the traffic distribution under the various cases. It is quite possible that further fine-tuning to reduce costs could have been found, or that a better allocation of the capacity might have been found. An anomalous situation does arise in the data for large jobs (20 MB) in that the centralized case seems more cost-effective. This is probably due to the fact that the bulk of the job traffic arises in the eastern cities, and, by moving computers away from the central moment of traffic sources, costs have been increased. Further investigation of this aspect of network optimization is indicated. Frank and Frisch [7] and Martin [11] have indicated approaches to the problem for communication nets. These, in conjunction with resource-allocation algorithms, should provide fairly ready answers to optimal location of processing centers.

There are, of course, several other arguments against complete centralization besides relative cost-effectiveness for throughput. The most relevant of these is the relative vulnerability of a completely centralized facility to the effects of failure of processing or transmission equipment (reliability impacts), environmental effects, such as blackouts or brownouts of electrical power, inclement weather, sabotage, or hostile action. On the other hand, larger processors frequently have other advantages, such as faster, more powerful peripheral equipment as well as superior and more powerful central processing units, instruction repertoires, and memories. Larger computers often have more powerful operating systems and programming languages available to them. Facility operation and maintenance costs of a few central facilities will (or may) also be lower, replacement parts are more easily handled, and record keeping and administration are made easier. Nonetheless, the opportunities for load balancing, the superior failsafe capabilities of alternative locations, and the opportunity for specialization of some computers for specific kinds of jobs with resultant increases in efficiency seem to bolster our general finding that semi-distributed computing provides a superior operation.

3. CONFIGURATION ANALYSIS

The second set of experiments was focused on the configuration and communications aspects of networking. In the first set of experiments, the network topology was fixed, and computation and message properties varied. Here, the goal was to configure a network based on articulation, reliability, and cost-effectiveness. By following several policies of link deletion from a completely connected configuration, the most cost-effective topology was derived under various constraints of articulation level. The cost-effectiveness measures of the resulting configurations are compared here with those of ring and star topologies. Another goal was to find the sensitivity between optimization with respect to topology and the communications traffic input data as well as the performance criteria.

The framework of these experiments was more restrictive than that of the first set. Eight nodes were selected in the following cities:

San Francisco	Boston
Los Angeles	New York
Chicago	Philadelphia
Detroit	Washington, D.C.

Initially, a fully connected network topology was assumed, so that any two sites could communicate directly. Line capacity was set, as before, at 50kb. No computation was done at any node, so that the computation processing characteristics were deleted.

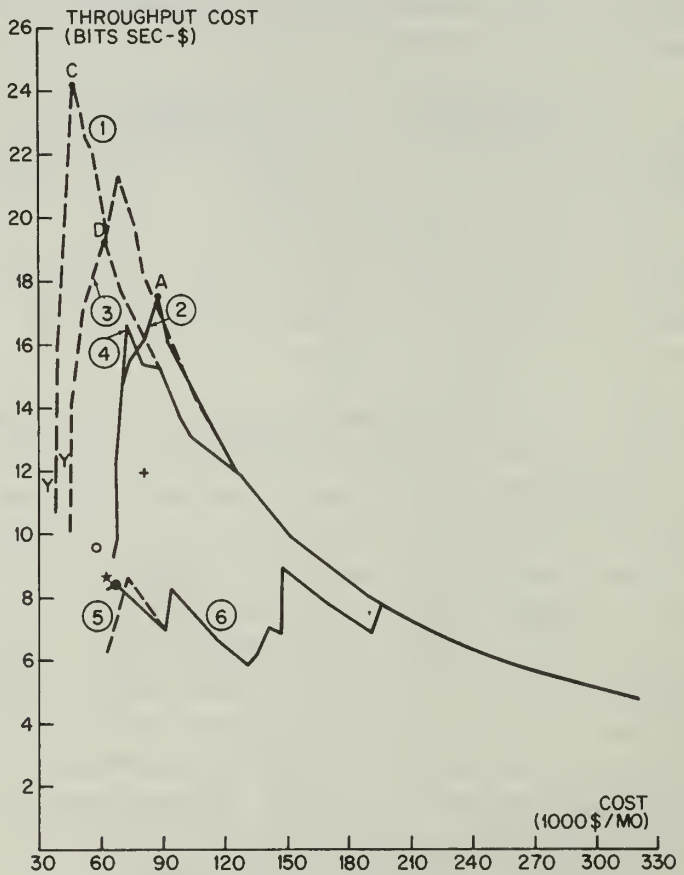


FIGURE 4. Configuration throughout cost-effectiveness.

The message size was set at 2,000 bits, and the packet size at 1,000 bits. No acknowledgment messages were assumed. Two job-arrival matrices were formed on two bases. The first was the distance-population formula of the first set of experiments. The second was a symmetric traffic matrix where an equal number of jobs was sent between any two nodes.

Costs were based on line costs, and reliabilities were based on line reliabilities. The beginning topology for all experiments was a completely connected one in which there was a direct connection between every two centers. Links were removed individually in a sequential manner. Two measures of cost-effectiveness were employed. One is based on throughput factored by cost. The curves in Figure 4 are based on this criterion. In this figure, cost is graphed versus bits/second per dollar cost.

Figure 5 is based on the cost-effectiveness measure of response time. Here the workload is a constant, so that cost is plotted versus a constant (10^5) divided by the product of mean total response time and monthly cost.

The topology-determination process of successive link removal through cost-effectiveness may be compared to the results achieved by *a priori* network specification. A topology can be prespecified, and then the links can be constructed on the basis of either geographic or message traffic considerations. Two common topologies are a ring and a star. A ring is the configuration with a minimum number of links that gives articulation level two, while a star has a central site and all other sites are connected only to the central site. Another method is to use minimal spanning trees. Distance or link values could be based on geography or traffic. Yet another method is to link each center to the two other centers with

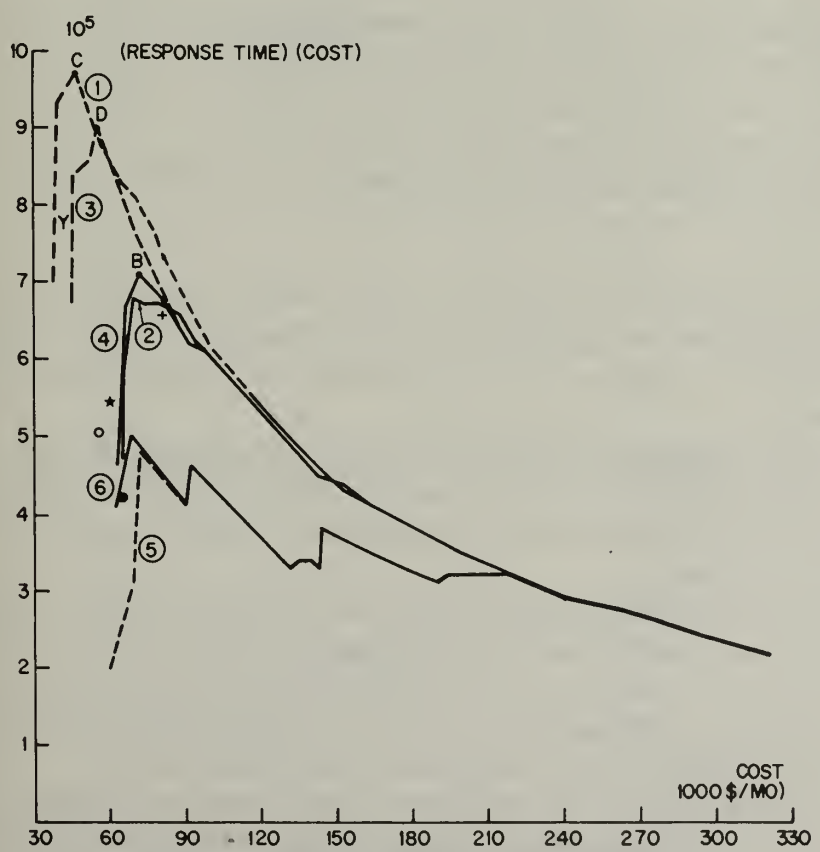


FIGURE 5. Configuration response time cost-effectiveness.

which it has the highest message traffic. These standard configurations are displayed in Figure 6, and their cost-effectiveness is given in Figures 4 and 5, using the following symbols:

ring—traffic	●
ring—geography	O
star—center at traffic center	*
min. spanning tree—traffic	$\underline{Y}$
min. spanning tree—geography	Y
heaviest communicating neighbors	+

In Figures 5 and 7, the following numerical code for the cities is used.

1 Los Angeles	5 Washington, D.C.
2 San Francisco	6 Philadelphia
3 Detroit	7 New York
4 Chicago	8 Boston

The curves in Figures 4 and 5 are labeled according to the following code:

Label	*Description*
①	Link removal by cost-effectiveness, articulation level 1
②	Link removal by cost-effectiveness, articulation level 2
③	Link removal by least loads, articulation level 1
④	Link removal by least loads, articulation level 2
⑤	Uniform traffic link removal, articulation level 1
⑥	Uniform traffic link removal, articulation level 2

Link removal by cost-effectiveness means that, at each step, the link that is least cost-effective in terms of throughput per unit cost is removed. Articulation level 2 means that the constraint was applied and that the network had to possess link articulation level 2 at each stage of link removal, including the final network.

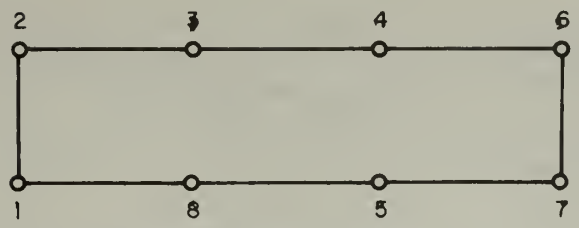
Another method of link removal was based on removal of the least-loaded link at a given stage.

Uniform traffic link removal refers to the removal of the least loaded link based on uniform traffic statistics. However, cost-effectiveness and the graph values are based on traffic which, is dependent upon population and distance.

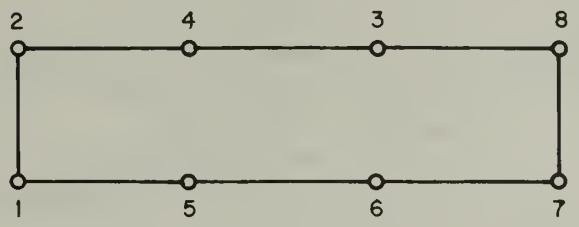
In Figures 4 and 5, the points at which some of the curves attain maximum value are of interest. Some of these have been labeled, and the corresponding configurations are given in Figure 7.

Two conclusions are evident. First, note that topology determination based on cost-effective and least-loaded considerations yield far better throughput cost-effectiveness (see Figure 4) than any standard topology assumed a priori without consideration of communication lines.

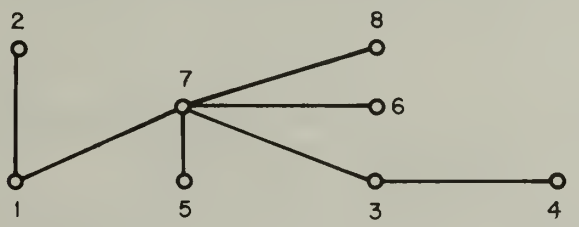
The curves also reveal that assuming a uniform traffic matrix does no better—in fact, poorer—than standard least-cost topologies. Hence, although superior when good information exists on traffic loads, the methodology is quite sensitive to inaccuracies in traffic estimates. Quite similar results are



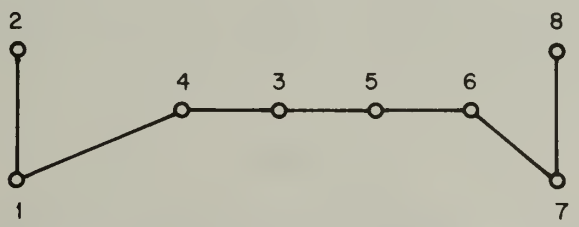
RING CONFIGURATION BASED ON TRAFFIC



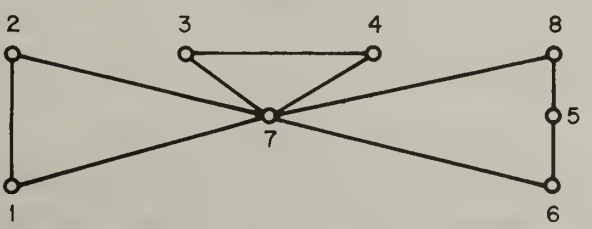
RING CONFIGURATION BASED ON GEOGRAPHY



MINIMAL SPANNING TREE CONFIGURATION BASED ON TRAFFIC

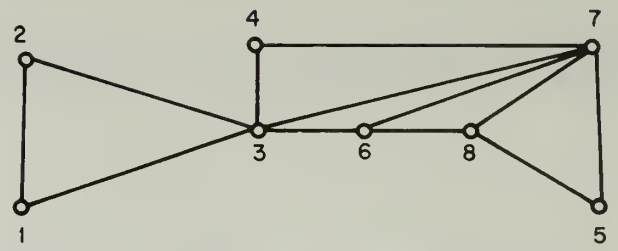


MINIMAL SPANNING TREE CONFIGURATION BASED ON GEOGRAPHY

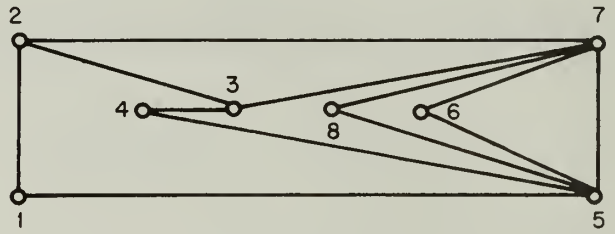


CONFIGURATION BASED ON THE HEAVIEST COMMUNICATING NEIGHBORS

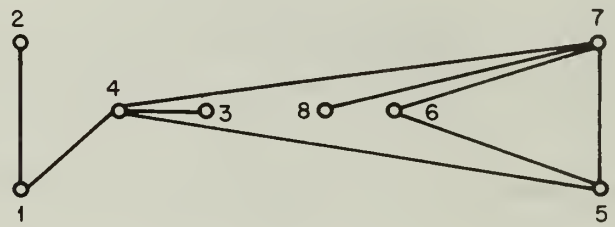
FIGURE 6. Standard configurations.



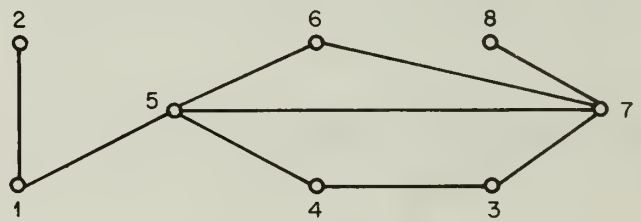
MOST THROUGHPUT COST EFFECTIVE WITH ARTICULATION LEVEL 2 - POINT A



MOST RESPONSE TIME COST EFFECTIVE WITH ARTICULATION LEVEL 2 - POINT B



MOST COST EFFECTIVE WITH ARTICULATION LEVEL 1 - LINK REMOVAL BY COST EFFECTIVENESS - POINT C



MOST COST EFFECTIVE WITH ARTICULATION LEVEL 1 - LEAST LOAD LINK REMOVAL - POINT D

FIGURE 7. Most cost-effective configurations.

achieved for response-time effectiveness, except that ranking node interconnections on probable load levels yields fairly good results (see points Y for a minimally articulated net and + for a two-articulated net in Figure 5. Thus, if traffic estimates are only good enough to establish probable ranks, then information will still yield better cost-effective performance than ignoring traffic patterns altogether.

The second conclusion is that the optimum configuration depends on the cost-effectiveness measure and the link-removal process. This is supported in the figures by comparing the maximum value with the constraint of articulation level two. In Figure 4, this point is labeled A and is attained by link removal based on cost-effectiveness. In Figure 5, however, the most cost-effective point is a different point (labeled B) and is obtained by the least-load link-removal process. These two configurations are quite different, as is revealed by comparing Figures 7(a) and 7(b). Since point A in Figure 4 (point B in Figure 5) corresponds to optimization by throughput (response time in Figure 5), throughput and response time are not necessarily optimized in the same configuration.

To consider different link-removal methods with or without the link articulation level being two, it is sufficient to examine either Figure 4 or Figure 5. The sensitivity of the maximum value attained to the link-removal method is resolved in the distinct curves and points where a maximum value is attained. Thus, it would be necessary to have several methods of link removal on hand for an automated process.

Another conclusion is that all optimization removal procedures depend on the traffic statistics and, in particular, the job-arrival matrix, which gives the number of jobs being sent between any two centers. This can easily be seen in both Figures 4 and 5 by examining the uniform-traffic link-removal curves. In both figures, the curves follow a zig-zag path and are dominated by the least-loaded link-removal cases. This is due to the experiments' being structured by one value of a parameter and evaluated on the basis of another value of the same parameter. Some of the sensitivity to message traffic, as a referee comments, is due to the fixed minimum path routing.

As might be expected, the most cost-effective networks are those that have articulation level one. This is seen in both Figures 4 and 5, where points C dominate points A and B. Furthermore, the configuration in Figure 4 and the traffic statistics based on population and distance reveal that the most cost-effective network configurations are those that are one-connected in remote or light traffic areas.

However, although level-one networks are more desirable from a cost-effectiveness point of view, they are not desirable when reliability is a consideration. A measure of reliability is the expected number of node pairs that will communicate. The failure rate can then be defined as the probability that a pair of nodes will not be able to communicate at a given time. With these definitions, consider the example of Figure 8. In the first network, there are 13 links, the articulation level is 2, and the failure rate

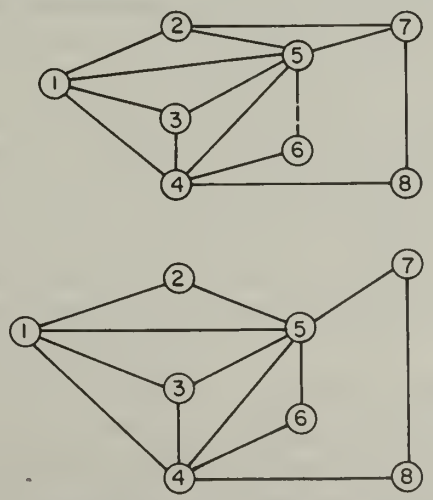


FIGURE 8. Sample networks—reliability and articulation level.

is 0.00012. However, in the second network the number of links is 12, the articulation level is 1, and the failure rate is 0.00681. In this particular example, by deleting the link from node 2 to node 7, the failure rate increased by a factor of 57. This is just one example of the articulation level. The exact dependence and increase in failure rate would depend on the network structure.

To summarize this section, we note that articulation level 1 is desirable, but not possible from a reliability point of view. Furthermore, a link-removal method such as is described here is better than a standard configuration because it is more sensitive to the structure of the network. However, to find the most cost-effective configuration, several methods of link elimination must be tried. When performing the analysis, statistics as close to the estimated or actual traffic as possible must be used.

REFERENCES

[1] *Keydata Computer Characteristics Review* (Keydata Corp., Waterton, Mass. 11, 1971).

[2] Auerbach Computer Characteristics Digest, Auerbach Info. Inc., Philadelphia (June 1972).

[3] Alter, R. and B. P. Lientz, "Applications of a Generalized Combinatorial Problem of Smirnov," Nav. Res. Log. Quart. **16**, 543–547 (1969).

[4] Alter, R. and B. P. Lientz, "A Note on a problem of Smirnov; A Graph Theoretic Interpretation," Nav. Res. Log. Quart. **17**, 407–408 (1970).

[5] Cady, George, "Computation and Communication Trade-Off Studies: An Analytical Model of Computer Networks," Proc. WESCON Conf. (Sept. 1972).

[6] Frank, H., *Minimum Link Traffic Routing*, unpublished (1970).

[7] Frank, H. and I. T. Frisch, *Communication, Transmission, and Transportation Networks* (Addison-Wesley Publishing Co., Reading, Mass., 1971).

[8] Kleinrock, L., *Communication Nets: Stochastic Message Flow and Delay* (McGraw-Hill, New York, 1964).

[9] Kleinrock, L., "Analysis and Simulation Methods in Computer Network Design," AFIPS Conf. Proc. **36**, 569–579 (1970).

[10] Lientz, B. P., "Combinatorial Problems in Communication Networks," *Combinatorics* (Academic Press, New York, 1972).

[11] Martin, James, *Telecommunications and the Computer* (McGraw-Hill, New York, 1972).

A GENERALIZED EUCLIDEAN PROCEDURE FOR INTEGER LINEAR PROGRAMS

Tyronza R. Richmond
Graduate School of Business
Howard University
Washington, D.C.

and

Arunachalam Ravindran
School of Industrial Engineering
Purdue University
Lafayette, Indiana

ABSTRACT

This paper investigates a new procedure for solving the general-variable pure integer linear programming problem. A simple transformation converts the problem to one of constructing nonnegative integer solutions to a system of linear diophantine equations. Rubin's sequential algorithm, an extension of the classic Euclidean algorithm, is used to find an integer solution to this system of equations. Two new theorems are proved on the properties of integer solutions to linear systems. This permits a modified Fourier-Motzkin elimination method to be used to construct a nonnegative integer solution. An experimental computer code was developed for the algorithm to solve some test problems selected from the literature. The computational results, though limited, are encouraging when compared with the Gomory all-integer algorithm.

1. INTRODUCTION

Since 1958, several algorithms have been proposed for solving the general-variable integer programming problem. Although most of these methods are mathematically finite, their computational performances are oft-time erratic and undependable. Today, some 13 years after Gomory [10] introduced his first integer programming algorithm, there still exists a real need for dependable computational methods for solving integer linear programs. Dantzig [9], Balinski [1, 2], and most recently Cushing [8] have shown that a number of problems of both theoretical and practical significance can be formulated as integer linear programs. However, before integer linear programming can be seriously considered as a practical tool or alternative method for solving these significant problems, integer programming algorithms must be developed with increased computational power—computational power possibly comparable to that of the simplex algorithm. Although it is unlikely that an integer programming algorithm with this desired computational power is forthcoming in the near future, different and varied solution procedures should and must be studied.

In this report, a new procedure for solving the pure integer programming problem is investigated. The resulting algorithm is a composite approach employing a generalization of the Euclidean algorithm and a modified Fourier-Motzkin elimination method. Results of an experimental study to compare the method with Gomory's [11] all-integer algorithm are quite encouraging.

2. STATEMENT OF THE PROBLEM

A pure integer programming problem is a linear programming problem with the additional restriction that all the variables be integer-valued. Mathematically, the pure integer linear programming problem may be stated as

$$\text{(P1)} \qquad \begin{aligned} &\text{Minimize} && z = cx \\ &\text{subject to} && Ax = b \\ &&& x\text{: a nonnegative integer vector,} \end{aligned}$$

where A is an $(m \times n)$ matrix, b is an $(m \times 1)$ column vector, and c is an $(1 \times n)$ row vector. For the pure integer programming problem (P1) we shall assume, without any loss in generality, that the components of A, b, and c are integers.

The linear programming problem defined by (P1) without the integrality restriction will be referred to as the associated linear program. It is assumed that (P1) is feasible and bounded, and that the solution to the associated linear program is not a solution to (P1).

In general, problem (P1) is equivalent to finding the smallest integer value of z such that the following system has a feasible solution:

$$\text{(2.1)} \qquad \begin{aligned} -cx + z &= 0 \\ Ax \quad &= b \\ x \geqq 0 &\text{ and integer.} \end{aligned}$$

Let z_0 be the (integer) lower bound for the smallest value of z. This lower bound z_0 may be determined by solving the associated linear program of (P1) and finding the smallest integer equal to or exceeding the optimum objective function value provided by the simplex algorithm.

Then the original problem (P1) is equivalent to solving the following problem (P2):

Find the smallest value of $(z_0 + k)$ such that,

$$\text{(2.2)} \qquad Hx = d - (z_0 + k)e \qquad \text{for } k = 0, 1, 2, \ldots$$

$$\text{(2.3)} \qquad x \geqq 0$$

has a feasible integer solution, where

$$H = \begin{bmatrix} -c \\ A \end{bmatrix}, \qquad d = \begin{bmatrix} 0 \\ b \end{bmatrix}, \qquad e = \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}.$$

H is a $(m' \times n)$ matrix, while d and e are both $(m' \times 1)$ vectors, where $m' = m + 1$.

In its simplest form, our algorithm can be described in the following way. For $k=0$, determine if an all-integer solution exists for (2.2). Using that all-integer solution, attempt to construct a nonnegative integer solution satisfying (2.2) and (2.3). If no nonnegative solution exists for (2.2) and (2.3), then increase k by one and repeat the process. For the first value of k yielding a solution to (2.2) and (2.3), an optimum solution to problem (P1) is readily available. In the actual algorithm, it is shown how it is not necessary to solve (2.2) and (2.3) for every k. Instead, a solution to (2.2) and (2.3) can be constructed by applying our algorithm just once.

3. LINEAR DIOPHANTINE EQUATIONS

In this section, we discuss methods for solving systems of linear diophantine equations, such as (2.2). A linear diophantine equation is a linear equation whose solution is restricted to be integers. There exists a number of classical methods for solving systems of linear diophantine equations. Saaty [16] has provided an extensive survey of these classical approaches. Recently, a more direct and less cumbersome method has been proposed which may be considered as a generalization or extension of the classical Euclidean algorithm. In this section, we shall describe Rubin's [15] sequential algorithm, which can be used to find an all-integer solution to the system (2.2). The principal sources for this section have been Rubin [15], Blankinship [4], and Gonzalez-Zubieta [12].

Let g.c.d. (x, y, z) denote the greatest common divisor of the numbers x, y, z. It will be assumed that the numbers are integers and that we are referring to the greatest common divisor of their absolute values.

LEMMA 3.1: The single linear diophantine equation

$$a_1x_1+a_2x_2+\ .\ .\ .+a_nx_n=b_1$$

has an all-integer solution, if and only if,

$$\text{g.c.d. } (a_1, a_2, \ .\ .\ ., a_n)=\text{g.c.d. } (a_1, a_2, \ .\ .\ ., a_n, b_1),$$

in other words, the g.c.d. $(a_1, a_2, \ .\ .\ ., a_n)$ divides b_1.

PROOF: (See Wahi [20])

It is also well known that the g.c.d. $(a_1, a_2, \ .\ .\ ., a_n)$ is unchanged by performing elementary operations on the vector, where an elementary operation is defined as either changing the sign of one element, or adding or subtracting from element a_i an integral multiple of another element a_j $(j \neq i)$. By elementary operations, the vector $(a_1, a_2, \ .\ .\ ., a_n)$ can be transformed to a vector possessing just one nonzero element. Thus from Lemma 3.1 and the fact that the g.c.d. is invariant after elementary operations, single linear diophantine equations can easily be solved or it can be shown that no integer solution exists.

Now consider a system of linear diophantine equations of the form $Ax=b$ where A is an $(m \times n)$ matrix with $m \leqq n$. Associated with this system is the homogeneous system $Ax=0$. If the rank of matrix A is m, then it is well known that this homogeneous system has $(n-m)$ independent solutions. Without any loss of generality, we can assume that the $(n-m)$ independent solutions are all-integer solutions. This set of solution is called a "fundamental set," and the matrix containing these independent solution

vectors as columns is called a fundamental matrix for A. Thus, if F is a fundamental matrix $(n \times n-m)$ for the system $Ax=b$, then it follows that x^*+Fy is also an integer solution for $Ax=b$, then it follows that x^*+Fy is also an integer solution for $Ax=b$ for every integer vector y.

Rubin [15] has extended the procedure to solve a single diophantine equation to a systematic procedure for solving a system containing $m \geqq 1$ diophantine equations. The following constitutes the central idea of Rubin's sequential approach.

THEOREM 3.1 (Rubin [15]):

Suppose the first r equations $(1 \leqq r < m)$ of the $(m \times n)$ diophantine system $Ax=b$ are solvable in integers. Let x^* be a particular integer solution and F be a fundamental matrix for the first r equations. Let a^{r+1} be the $(r+1)$st row of A and b_{r+1} be the $(r+1)$st component of the right hand side. Then there exists an integer solution to the first $(r+1)$ equations if and only if the single equation

$$a^{r+1}x^*+a^{r+1}Fz=b_{r+1}$$

is solvable in integer z_j's. In addition, if the vector z^* solves the above single equation, then x^*+Fz^* solves the first $(r+1)$ equations. Furthermore, if $a^{r+1}F=0$ and $b_{r+1}-a^{r+1}x^*=(\neq)0$, the equation $(r+1)$ is redundant (inconsistent).

Sequential Algorithm for Diophantine Equations

Theorem 3.1 in conjunction with Lemma 3.1 provides a theoretical basis for the development of the sequential algorithm for solving a diophantine system. It is recommended that the reader refer to the numerical example given in appendix A side by side for a better understanding of the general steps of the algorithm given in this section.

The sequential algorithm constructs a solution vector x for the $(m \times n)$ diophantine system $Ax=b$, by performing column operations on the matrix A. An $(n \times n)$ identity matrix shall be appended to the A matrix to keep track of the column operations. Denote by W, the $(m+n) \times n$ matrix $\begin{bmatrix} A \\ I \end{bmatrix}$. As the column operations are performed, the resulting matrix shall still be referred to as W. The elements of matrix W will be referred to as w_{ij} and we shall define $w_{1,-1}=0$. The general steps of the sequential algorithm can be described as follows:

STEP I: Set $i=0$, $r=0$.

STEP II: Set $i=i+1$. If $i > m$, go to STEP VII. Otherwise to STEP III.

STEP III: Let w_{ij} be the nonzero element of the smallest absolute value in the set $\{w_{ip}\}$ where $p=i-r, i-r+1, \ldots, n$. If $w_{ij} \neq 0$, then go to STEP IV. Instead, if all elements of the set are zero and

(a) $w_{i1}+\ldots+w_{i,i-r-1}=b_i$, then the system is redundant, in which case set $r=r+1$ and go to STEP II.

(b) $w_{i1}+\ldots w_{i,i-r-1} \neq b_i$, then the system is inconsistent, in which case stop.

STEP IV: Let w_{ik} be any other nonzero member of the set, $k \neq j$. If no such element exists, switch columns $i-r$ and j and go to STEP VI. Otherwise go to STEP V.

STEP V: Let $g=[w_{ik}/w_{ij}]=$ the greatest integer not exceeding w_{ik}/w_{ij}. Substract g times the jth column from the kth column. If the new $w_{ik}=0$, go to STEP IV. If the new $w_{ik} \neq 0$, switch j and k and repeat STEP V.

STEP VI: Compute $\lambda_i = b_i - w_{i1} - w_{i2} - \ldots - w_{i,i-r-1}$.

(a) If $w_{i,i-r}$ divides λ_i, let $t_i = \lambda_i / w_{i,i-r}$. Multiply column $(i-r)$ by t_i. Then go to STEP II.

(b) If $w_{i,i-r}$ does not divide λ_i, then the system does not have an all-integer solution, in which case stop.

STEP VII:

$$\text{Set } z_i = \begin{bmatrix} w_{m+1,i} \\ w_{m+2,i} \\ \cdot \\ \cdot \\ \cdot \\ w_{m+n,i} \end{bmatrix} \quad \text{for } i = 1, 2, \ldots, m-r.$$

Let $x^* = z_1 + z_2 + \ldots + z_{m-n}$

and

$$F = \begin{bmatrix} w_{m+1,m-r+1} & \ldots & w_{m+1,n} \\ w_{m+2,m-r+1} & \ldots & w_{m+2,n} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ w_{m+n,m-r+1} & \ldots & w_{m+n,n} \end{bmatrix}$$

Then x^* is an all-integer solution to $Ax = b$ and F is a fundamental matrix for A. Thus, for every integer vector y, $x^* + Fy$ is also an integer solution to $Ax = b$.

4. GENERALIZED EUCLIDEAN ALGORITHM

In section 2 it was shown that solving the integer program (P1) is equivalent to solving the following problem (P2): find the smallest value of $(z_0 + k)$ such that

$$Hx = d - (z_0 + k)e \qquad \text{for } k = 0, 1, 2, \ldots \tag{4.1}$$

$$x \geqq 0 \tag{4.2}$$

has a feasible integer solution.

For a particular value of k, the sequential algorithm presented in section 3 can be used to construct a particular integer solution vector $x(k)$, and a fundamental matrix F, for the $(m' \times n)$ diophantine system given by (4.1). If $x(k)$ is not nonnegative then one may try to find an all integer solution vector y for the following inequality system:

$$x(k) + Fy \geqq 0. \tag{4.3}$$

If there exists an all integer solution y^* for (4.3), then $x^* = x(k) + Fy^*$ solves (4.1) and (4.2), and x^* is an optimal solution to the integer program (P1). If no integer solution y exists for (4.3), then k could be increased by one, and the resulting system defined by (4.1) could be solved again by the sequential algorithm to find $x(k+1)$. This intuitive approach could be continued until for some value of k, Equations (4.1) and (4.2) would have nonnegative integer solutions.

In this section, we will present two basic theorems which will show that it is not necessary to examine successive values of k for (4.1) and that (4.3) need not be solved more than once. As a matter of fact, Theorems 4.1 and 4.2, provide the basis for the development of the Generalized Euclidean algorithm presented in this paper for the solution of integer programming problems. A discussion of the modified Fourier-Motzkin procedure to solve (4.3) and the detailed steps of the Generalized Euclidean Algorithm conclude this section.

THEOREM 4.1: Let k_0 and $k_1 > k_0$ be the two smallest nonnegative values of k such that (4.1) has all-integer solutions. Then the system (4.1) has all integer solutions only for

$$k \epsilon K = \{k | k = k_0 + tm_0,\ t \text{ a nonnegative integer}\}$$

where $m_0 = k_1 - k_0$.

PROOF: (by mathematical induction on the number of equations)

CASE 1: Consider the one equation system ($m' = 1$) of (4.1):

$$h_{11}X_1 + h_{12}X_2 + \ldots + h_{1n}X_n = d_1 - (z_0 + k)e_1$$

Let

$$\alpha = \text{g.c.d. } (h_{11}, h_{12}, \ldots, h_{1n}).$$

Since (4.1) has integer solutions for $k = k_0$ and $k = k_1$, it is known from Lemma 3.1 that

(*i*) $[d_1 - (z_0 + k_0)e_1]/\alpha$ is an integer;
(*ii*) $[d_1 - (z_0 + k_1)e_1]/\alpha$ is an integer.

Substituting $k_1 = k_0 + m_0$ in (ii), we get

$$[d_1 - (z_0 + k_0 + m_0)e_1]/\alpha \quad \text{is an integer.}$$

This implies that $(m_0 e_1)/\alpha$ is an integer. Thus, for any integer t,

$[d_1 - (z_0 + k_0 + tm_0)e_1]/\alpha$ is an integer and (4.1) has an integer solution for

$$k \epsilon K \quad \text{for every integer } t.$$

On the other hand, suppose there exists no integer t such that $k = k_0 + tm_0$. Then for some positive integer $s < m_0$, k can be represented as

$$k = k_0 + tm_0 + s < k_0 + (t+1)m_0.$$

Since (4.1) has no integer solution between k_0 and k_1, it follows that se_1/α is not an integer. Hence, no integer solution exists for (4.1) when $k = k_0 + tm_0 + s$.

CASE 2: Now assume that the theorem is true for $m' = r > 1$ and prove that it is true for $m' = r+1$.

By our assumption, let x^r be an integer solution and F^r denote a fundamental matrix for the first r equations. From Theorem 3.1, an integer solution exists for the first $(r+1)$ equations only if the single equation

$$h^{r+1}F^r z = d_{r+1} - (z_0 + k)e_{r+1} - x^r h^{r+1} \tag{4.4}$$

has an integer solution. Since we have proved the theorem for the one equation system (case 1), it follows that Theorem 4.1 is true for Equation (4.4) also. This implies that if (4.4) has integer solutions for $k = k_0$ and $k = k_1$, then it has integer solutions for only those $k = k_0 + tm_0$ for all positive integer t. This in conjunction with Theorem 3.1, proves the result for case 2.

The above theorem showed that system (4.1) has integer solutions only for some restricted values of k. The following theorem will exhibit a relation between the various integer solutions to (4.1) for different values of k.

THEOREM 4.2: Let k_0 and $k_1 > k_0$ be the two smallest nonnegative integer values of k for which (4.1) has all-integer solutions, denoted by $x(k_0)$ and $x(k_1)$, respectively. Then, there exists an all integer solution vector $x(k)$ for (4.1) of the form

$$x(k) = x(k_0) - (k - k_0)f \tag{4.5}$$

only for those $k = k_0 + tm_0$ where f is a constant integer vector, called the difference vector, $m_0 = k_1 - k_0$ and t is any positive integer.

PROOF: The proof is rather involved and for brevity sake, it is not presented here. Interested readers may refer to the original source [14].

From Theorems 4.1 and 4.2, we note that in order to solve problem (P2), there is no need to solve (4.1) for each consecutive value of k. After finding the integer solutions to (4.1) for the two smallest values of k (k_0 and k_1), we can construct the difference vector

$$f = -[x(k_1) - x(k_0)]/(k_1 - k_0).$$

This in turn gives the solutions to other values of k, using the relation (4.5).

$$x(k) = x(k_0) - (k - k_0)f.$$

Thus, solving problem (P2) is greatly simplified as one has to solve (4.1) at most twice only, and then use relation (4.5) to solve the system (4.3) in just one step.

Thus, once $x(k_0)$ and f are known, then problem (P2) is equivalent to finding the smallest nonnegative integer k such that

$$x(k_0) - (k - k_0)f + Fy \geqq 0 \tag{4.6}$$

has an integer solution vector y.

Now we shall describe the Fourier-Motzkin elimination method to solve (4.6). This description is based primarily on the account provided by Wahi [20].

Fourier-Motzkin Elimination Method

The Fourier-Motzkin elimination method has been successfully used to solve small linear programming problems. The inequalities defined by (4.6) can be rewritten as

$$f_{i1}y_1+f_{i2}y_2+\ldots+f_{ir}y_r-f_ik \geqq p_i \qquad \text{for } i=1, 2, \ldots n, \tag{4.7}$$

where $p_i=-x_i(k_0)-k_0f_i$.

Note: The dimension of matrix F is considered as $(n\times r)$ where $r=(n-\text{rank of matrix } A)$.
To find an integer solution vector y for (4.7) for the smallest nonnegative integer k, we apply the Fourier-Motzkin method as follows:

(i) Choose variable y_1, and partition the set of n inequalities into three groups:

$$k_1=[s: f_{s1}>0]$$

$$k_2=[t: f_{t1}<0]$$

$$k_3=[u: f_{u1}=0]$$

(ii) Eliminate y_1, and obtain resulting system of linear inequalities as shown below:

(a) Every inequality in the set k_1 can be written as

$$p_s/f_{s1}-\sum_{j=2}^{r}(f_{sj}/f_{s1})y_j+kf_s/f_{s1} \leqq y_1 \qquad \text{for } s\epsilon k_1.$$

(b) Every inequality in the set k_2 can be written as

$$y_1 \leqq p_t/f_{t1}-\sum_{j=2}^{r}(f_{tj}/f_{t1})y_j+kf_t/f_{t1} \qquad \text{for } t\epsilon k_2.$$

(c) Those inequalities in k_3 can be written as

$$\sum_{j=2}^{r}f_{uj}y_j-kf_u \geqq p_u \qquad \text{for } u\epsilon k_3.$$

(iii) For every $s\epsilon k_1$ and $t\epsilon k_2$

$$p_s/f_{s1}-\sum_{j=2}^{r}(f_{sj}/f_{s1})y_j+kf_s/f_{s1} \leqq p_t/f_{t1}-\sum_{j=2}^{r}(f_{tj}/f_{t1})y_j+kf_t/f_{t1}.$$

The resulting system of inequalities with one less unknown (y_1) is

$$\sum_{j=2}^{r}(f_{sj}/f_{s1}-f_{tj}/f_{t1})y_j+k(f_t/f_{t1}-f_s/f_{s1}) \geqq (p_s/f_{s1}-p_t/f_{t1}), \qquad \text{for all } s\epsilon k_1 \text{ and } f\epsilon k_2.$$

$$\sum_{j=2}^{r}f_{uj}y_j-kf_u \geqq p_u \qquad \text{for } u\epsilon k_3.$$

It should be noted here that the basic Fourier-Motzkin elimination method can result in an increased number of inequalities. If the sets k_1 and k_2 contain $n/2$ and $n/2$ elements each, then the number of inequalities can grow at an exponential rate. Cook and Cooper [7] have proposed a procedure for maintaining the number of inequalities constant, while Bradley and Wahi [5] have shown how the Fourier-Motzkin elimination method can be improved by structuring the coefficient matrix.

For our present discussion, we shall use only the basic elimination method described above, which can be continued to the other variables $y_2, \ldots y_r$. Eventually, a system of inequalities will be obtained with one (last) unknown variable. This last set establishes absolute bounds on the variable k.

Cook and Cooper [7] have modified the elimination method from hereon, to determine integer solutions to systems of linear inequalities. This will be used to determine the smallest nonnegative integer k for which an integer solution y exists for (4.7). The modified elimination method proceeds as follows:

Denote $\lfloor a \rfloor$ = the greatest integer $\leqq a$

$\lceil a \rceil$ = the smallest integer $\geqq a$.

First eliminate the variables of (4.7) until only k appears in the inequality set. The last inequality set provides lower (L_k) and upper (U_k) bounds on k. Set $\lceil L_k \rceil = \text{Max } (0, \lceil L_k \rceil)$ to ensure that only nonnegative values of k are considered.

(a) If $\lceil L_k \rceil > \lfloor U_k \rfloor$, then no integer solution exists for (4.7).

(b) If $\lceil L_k \rceil \leqq \lfloor U_k \rfloor$, then set $k = \lceil L_k \rceil$

Substitute the value of k in the inequality set involving only y_r and k. This yields conditional lower (L_r) and upper (U_r) bounds on y_r. If $\lceil L_r \rceil \leqq \lfloor U_r \rfloor$, set $y_r = \lceil L_r \rceil$ and backtrack another step to find conditional bounds on y_{r-1}. This process is continued until one of the two following cases occur:

(i) The substitution continues to find bounds on y_1, without ever encountering $\lceil L_j \rceil > \lfloor U_j \rfloor$ at any stage of the backtracking. Then a feasible integer solution to (4.7) is

$$y_j = \lceil L_j \rceil \qquad \text{for } j = 1, 2, \ldots, r \text{ and}$$

$$k = \lceil L_k \rceil.$$

This in turn gives the optimum solution to the Integer Program (P1).

(ii) For some $j (1 \leqq j \leqq r)$, $\lceil L_j \rceil > \lfloor U_j \rfloor$. This implies that $y_i = \lceil L_i \rceil$ for $i = j+1, \ldots r$ and $k = \lceil L_k \rceil$ are inconsistent with the original inequalities. If $\lceil L_{j+1} \rceil < \lfloor U_{j+1} \rfloor$, then increment L_{j+1} by one and proceed to find new bounds on y_j. If $\lceil L_{j+1} \rceil$ cannot be increased (i.e., $\lceil L_{j+1} \rceil = \lfloor U_{j+1} \rfloor$) then backtrack further to find the smallest $i (j+1 < i \leqq r)$, such that $\lceil L_i \rceil < \lfloor U_i \rfloor$. If one exists, increase that $\lceil L_i \rceil$ by one and proceed from there. If none exists, then increase $\lceil L_k \rceil$ (if possible) by $(k_1 - k_0)$ and proceed. Instead, if $\lceil L_k \rceil + (k_1 - k_0) > \lfloor U_k \rfloor$, then this implies no feasible integer solution exists to the original inequalities (4.1) and (4.2); in other words, the given Integer Program does not have an optimum solution.

Now we describe the detailed steps of our Generalized Euclidean Algorithm below.

Generalized Euclidean Algorithm

The algorithm outlined here is for the integer programming problem (P1) defined in section 2.

STEP I: Use the simplex method to solve the linear programming problem associated with problem (P1). Denote the smallest integer greater than or equal to the optimal value of the associated linear program as z_0. If the linear programming solution is integral, then the problem is solved. Otherwise proceed to STEP II.

STEP II: Transform the given problem (P1) to the equivalent problem (P2). From (P2), the system of inequalities defined by (4.1) and (4.2) is obtained.

STEP III: Use Rubin's sequential algorithm given in section 3 to find $x(k_0)$ and F, an integer solution and a fundamental matrix, respectively, for the diophantine system (4.1) when $k=k_0$.

STEP IV: If possible, use the sequential algorithm to find $x(k_1)$, an integer solution for (4.1) when $k=k_1$ where k_1 is the smallest integer greater than k_0 yielding an integer solution for (4.1). If k_1 cannot be determined, set $x(k_1)=x(k_0)$.

STEP V: Construct the difference vector

$$f=-[x(k_1)-x(k_0)]/(k_1-k_0).$$

STEP VI: Use the modified Fourier-Motzkin elimination method to solve

$$x(k_0)+Fy-(k-k_0)f \geqq 0$$

for smallest integer $k(k \geqq 0)$ and integer vector y.

STEP VII: If the solutions obtained in STEP VI were k^* and y^*, then the optimal solution to the integer program is given by

$$x^*=x(k_0)+Fy^*-(k^*-k_0)f$$

and the optimum value is (z_0+k^*).

REMARKS:

(i) In STEP III, k_0 is used to represent the smallest value of $k \geqq 0$ yielding an all-integer solution to (4.1). For the test problems discussed in the next section k_0 came out to be zero in each case.

(ii) In STEP IV, it is possible theoretically for (4.1) not to possess an all-integer solution for any value of $k > k_0$. In such case, there is no need for the difference vector f and we apply STEP VI to the system

$$x(k_0)+Fy \geqq 0$$

to find an integer y. If there exists no integer solution y, then (P1) has no solution.

(iii) In appendix B, a complete numerical example illustrating the various steps of the algorithm is presented.

5. COMPUTATIONAL EXPERIENCE

The real worth of any mathematical algorithm lies in its computational feasibility and strength. An experimental FORTRAN IV code for the Generalized Euclidean Procedure for Integer Program (GEPIP) has been developed. The object code and data are all-in-core. It should be noted that only the basic Fourier-Motzkin elimination method is used.

Some 28 test problems were selected from the literature to test the computational efficiency of the GEPIP algorithm. Comparison was made with the IPSC code. The IPSC code is an all FORTRAN version of the Gomory all-integer algorithm developed by Trauth and Woolsey [19].

The 28 test problems may be classified as:

(i) nine fixed charge problems from Haldi [13],

(ii) nine allocation (knapsack) problems from Trauth and Woolsey [18],

(iii) one graph theory problem from Trauth and Woolsey [18],

(iv) two IBM test problems from Haldi [13],

(v) three stopped simplex problems from Thompson [17],

(vi) two Gomory problems [11], and

(vii) two Cook problems [6].

The computational results for the nine fixed charge problems are presented in Table 1, while Table 2 contains the results for the nine allocation problems. The results for the remaining ten problems are in Table 3. In each table, columns 3 and 4 give the size of each problem. Column 5 provides the solution value of the associated linear program and column 6, the solution value of the integer linear program. Columns 7 and 8 contain the computational times, in seconds, respectively for the Generalized Euclidean Procedure for Integer Programs (GEPIP), developed in this report, and the Gomory all-integer algorithm. Both algorithms were run on the Purdue University CDC 6500.

TABLE 1. *Fixed Charge Problems*

Problem No.	Source	m	n	LP z_0	IP z_0	GEPIP time	IPSC time
1	Haldi	4	9	8.787	7	0.118	0.257
2	Haldi	4	9	9.612	8	0.101	0.311
3	Haldi	4	9	11.812	10	0.129	0.508
4	Haldi	4	9	9.22	8	0.092	0.208
5	Haldi	6	11	88.61	76	3.380	37.508
6	Haldi	6	11	118.12	106	3.740	30.618
7	Haldi	4	9	88.61	76	32.670	.....
8	Haldi	4	9	118.12	106	32.253	.....
9	Haldi	6	12	12.0	9	0.368	0.638

TABLE 2. *Allocation Problems*

Problem No.	Source	m	n	LP z_0	IP z_0	GEPIP time	IPSC time
10	Trauth-Woolsey	11	21	49.5	48	0.882	1.600
11	Trauth-Woolsey	11	21	53.4	52	1.257	1.255
12	Trauth-Woolsey	11	21	57.04	55	2.047	1.230
13	Trauth-Woolsey	11	21	60.6	58	2.155	.981
14	Trauth-Woolsey	11	21	64.2	61	3.373	1.253
15	Trauth-Woolsey	11	21	67.8	66	1.335	1.123
16	Trauth-Woolsey	11	21	71.3	70	1.936	0.967
17	Trauth-Woolsey	11	21	74.7	72	3.911	1.099
18	Trauth-Woolsey	11	21	81.3	78	3.971	2.437

TABLE 3. *Selected Problems*

Problem No.	Source	m	n	LP z_0	IP z_0	GEPIP time	IPSC time
19	Trauth-Woolsey	8	14	2.0	2.0	0.105	0.155
20	Haldi	7	14	7.5	8	0.232	0.207
21	Haldi	7	14	5.75	7	0.500	0.266
22	Thompson	2	4	.50	100	0.059	0.210
23	Thompson	2	4	1097.7	1650	0.238	0.407
24	Thompson	2	4	2202.2	2750	0.241	0.485
25	Gomory	2	7	99.7	99	0.037	0.108
26	Cook	3	6	14.0	11	0.042	0.076
27	Gomory	3	6	19.0	19	0.033	0.08
28	Cook	4	8	4.3	3	0.055	0.118

The nine fixed-charge problems, developed by Haldi [13], have been used extensively as a test source for budding integer programming algorithms. The first four problems are similar and were solved rather easily by both algorithms. Note that for these first four problems, the optimal objective function values of the linear and integer programs are close. Problems 5 and 6 proved to be more difficult for both algorithms. Problems 5 and 6 are similar to problems 1 and 3, respectively, with two minor changes; the first change being two of the columns of the coefficient matrices and the right hand side vector have been scaled up by a factor of 10, and secondly, two additional constraints have been added restricting two of the variables to be binary. These changes significantly increase the computational difficulty. The Gomory algorithm required approximately 10 times as much time as did the GEPIP to construct the solutions. Problems 7 and 8, despite their apparent smallness are difficult problems to solve. Both problems were solved in excess of 30 seconds by use of the experimental computer code, GEPIP. For problems 7 and 8, the Gomory algorithm was stopped after exceeding 9,000 iterations (approximately 50 seconds) without converging to a solution. Trauth and Woolsey [18] have reported that the Gomory all-integer algorithm failed to reach a solution in 15,000 iterations for problems 7 and 8. Problem 9 provided no real difficulty for either algorithm.

The nine allocation problems were developed by Trauth and Woolsey [18], and are used to investigate the sensitivity of integer programming algorithms to a relatively minor change in the problem matrix. The nine allocation problems are zero-one knapsack problems and differ only in the non-trivial constraint. The results are given in Table 2. The minor changes had relatively minor effects on both algorithms. Although, the results are competitively close, the all-integer algorithm performed slightly better for these nine zero-one problems.

The results for the remaining selected problems are given in Table 3. For the majority of these 10 problems, the GEPIP performed better than the Gomory all-integer algorithm.

Although no general and far reaching conclusions can be drawn from the limited computational results, the results do indicate that the algorithm presented here is competitive and worthy of further study, consideration, and refinement.

6. CONCLUDING REMARKS

The bulk of the computational effort currently is devoted to the Fourier-Motzkin elimination method. Cook [6] has developed a more efficient version of the elimination method that can be studied for adaptation. Wahi [20] has shown how the Fourier-Motzkin elimination method can be improved immensely. The Wahi approach is based on triangularizing the coefficient matrix of the system of inequalities. The Wahi approach is currently limited to inequality systems where the coefficient system is square or almost square.

Currently, we require an initial estimate of z. At present, this estimate is obtained from the simplex method and the associated linear program. Based on the computational results presented here, problems where the values of the objective functions of the linear and integer programs are far apart at optimality are more difficult to solve. The approach presented here would be vastly improved by a better estimate for some problems. A better estimate perhaps could be obtained by employing the approximation ideas proposed by Biondi and Schmid [3].

In closing, it should be noted that Theorems 4.1 and 4.2 developed in this research effort have enhanced the future role of the Fourier-Motzkin elimination method in integer linear programming.

APPENDIX A

In this section, the solution of a three-equation diophantine system is considered to illustrate the mechanics of Rubin's sequential algorithm presented in section 3. In the example a (4×4) identity matrix is appended to the coefficient matrix and the entire matrix will be denoted by W. The solution vector x^* and the fundamental matrix F will be determined by performing column operations on W. As the column operations are performed the resulting matrices shall still be referred to as W. The elements of matrix W will be denoted by w_{ij}.

Example 1

$$\begin{aligned} 4x_1+5x_2+7x_3+2x_4&=11\\ 2x_1+7x_2-\ x_3+4x_4&=\ 7\\ x_1+2x_2+\ x_3+2x_4&=\ 2 \end{aligned}$$

NOTE: It is recommended that the reader refer to the general steps of the algorithm given in section 3 side by side.

Appending a (4×4) identity matrix, we get

$$W_{(7\times 4)}=\left[\begin{array}{rrrr} 4 & 5 & 7 & 2\\ 2 & 7 & -1 & 4\\ 1 & 2 & 1 & 2\\ \hdashline 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 \end{array}\right]$$

Perform the following sequence of column operations on W so as to reduce the first row vector (4, 5, 7, 2) to a vector (1, 0, 0, 0):

(i) Subtract two times column 4 from column 1. (Usually the element with the smallest absolute value in the row is chosen for eliminating the other nonzero elements.)

(ii) Subtract two times column 4 from column 2.

(iii) Subtract two times column 2 from column 4.

(iv) Subtract seven times column 2 from column 3.

(v) Interchange columns 1 and 2.

The above operations result in a transformed W matrix as given below:

$$W = \left[\begin{array}{rrrr} 1 & 0 & 0 & 0 \\ -1 & -6 & 6 & 6 \\ -2 & -3 & 15 & 6 \\ \hdashline 0 & 1 & 0 & 0 \\ 1 & 0 & -7 & -2 \\ 0 & 0 & 1 & 0 \\ -2 & -2 & 14 & 5 \end{array}\right]$$

The g.c.d. $(4, 5, 7, 2) = w_{11} = 1$. Equation 1 of Example 1 has an integer solution if and only if w_{11} divides $\lambda_1 = b_1 = 11$. Thus $t_1 = \lambda_1 / w_{11} = 11$.

Multiply column 1 of the above matrix by $t_1 = 11$ to get,

$$W = \left[\begin{array}{rrrr} 11 & 0 & 0 & 0 \\ -11 & -6 & 6 & 6 \\ -22 & -3 & 15 & 6 \\ \hdashline 0 & 1 & 0 & 0 \\ 11 & 0 & -7 & -2 \\ 0 & 0 & 1 & 0 \\ -22 & -2 & 14 & 5 \end{array}\right]$$

The first column of the appended matrix (below the dotted line), namely, $(0, 11, 0, -22)^T$ is an integer solution to the first equation of Example 1. The remaining (3×3) matrix is a fundamental matrix to the first equation. From Theorem 3.1, the first two equations of Example 1 has an integer solution if and only if the single equation

$$-11 - 6z_1 + 6z_2 + 6z_3 = 7 \tag{A.1}$$

has an integer solution. Note that the second row of the final W matrix is, in effect, equation (A.1).

Hence, (A.1) has an integer solution if and only if the g.c.d. $(-6, 6, 6)$ divides 18. By continuing the column operations on the W matrix, the g.c.d. $(-6, 6, 6)$ can be determined. Note that no operation is performed on the first column of W.

$$W = \left[\begin{array}{cccc} 11 & 0 & 0 & 0 \\ -11 & -6 & 0 & 0 \\ -22 & -3 & 12 & 3 \\ \hline 0 & 1 & 1 & 1 \\ 11 & 0 & -7 & -2 \\ 0 & 0 & 1 & 0 \\ -22 & -2 & 12 & 3 \end{array}\right].$$

Thus, $w_{22} = -6$ is the g.c.d. $(-6, 6, 6)$. Equation (A.1) has an integer solution if and only if $w_{22} = -6$ divides $\lambda_2 = b_2 - w_{21} = 7 - (-11) = 18$. Hence $t_2 = \lambda_2 / W_{22} = -3$. Multiply column 2 by $t_2 = -3$, and perform column operations on the last two columns of matrix W to determine the g.c.d. (12, 3). The following W matrix is then obtained:

$$W = \left[\begin{array}{cccc} 11 & 0 & 0 & 0 \\ -11 & 18 & 0 & 0 \\ -22 & 9 & 3 & 0 \\ \hline 0 & -3 & 1 & -3 \\ 11 & 0 & -2 & 1 \\ 0 & 0 & 0 & 1 \\ -22 & 6 & 3 & 0 \end{array}\right].$$

Thus, $w_{33} = 3$ is the g.c.d. (12, 3). Compute

$$\lambda_3 = b_3 - w_{31} - w_{32} = 2 + 22 - 9 = 15.$$

Hence $t_3 = \lambda_3 / w_{33} = 5$. Multiply column 3 by $t_3 = 5$ to get

$$W = \left[\begin{array}{rrrr} 11 & 0 & 0 & 0 \\ -11 & 18 & 0 & 0 \\ -22 & 9 & 15 & 0 \\ \hdashline 0 & -3 & 5 & -3 \\ 11 & 0 & -10 & 1 \\ 0 & 0 & 0 & 1 \\ -22 & 6 & 15 & 0 \end{array}\right]$$

An integer solution x^* to Example 1 may be obtained by summing the first three column vectors in the appended matrix (below the dotted line), and a fundamental matrix F for the system is given by the remaining columns of the appended matrix.

Thus

$$x^* = \begin{bmatrix} 0 \\ 11 \\ 0 \\ -22 \end{bmatrix} + \begin{bmatrix} -3 \\ 0 \\ 0 \\ 6 \end{bmatrix} + \begin{bmatrix} 5 \\ -10 \\ 0 \\ 15 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ -1 \end{bmatrix}$$

and

$$F = \begin{bmatrix} -3 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Furthermore, any $x = x^* + Fy$, for all integer y, is also an integer solution to Example 1.

APPENDIX B—SAMPLE PROBLEM

In this section, a sample problem is solved using the Generalized Euclidean Procedure for Integer Programs to illustrate the operations of the algorithm. The following numerical example is considered:

Example 2:

Maximize z

Subject to:

$$\begin{aligned} 2z+7x_1-\ 7x_2+x_3 &= 4 \\ 4z+9x_1-12x_2\ \ +x_4 &= 2 \\ -3z-7x_1+10x_2\ \ +x_5 &= 3 \\ -\ z-3x_1+\ 3x_2\ \ +x_6 &= 0 \\ x_1, \ldots, x_6 \geqq 0 & \\ x_1, \ldots, x_6, z \text{ integer.} & \end{aligned}$$

STEP I: Use the simplex method to solve the associated linear program ignoring the integer restrictions. From the simplex, $z_0=14$. Since we are maximizing, set $z=z_0-k=14-k$ where $k=0, 1, 2, \ldots$.

STEP II: Transform the given problem (P1) to the equivalent problem (P2) as follows:

Determine the smallest value of k such that

(B.1)
$$\begin{aligned} 7x_1-\ 7x_2+x_3 &= 4-2(14-k) \\ 9x_1-12x_2\ \ +x_4 &= 2-4(14-k) \\ -7x_1+10x_2\ \ +x_5 &= 3+3(14-k) \\ -3x_1+\ 3x_2\ \ +x_6 &= 0+1(14-k) \end{aligned}$$

for $k=0, 1, 2, \ldots$

(B.2) $$x_1, x_2, \ldots, x_6 \geqq 0$$

has a feasible integer solution.

STEP III: Use Rubin's sequential algorithm to find an integer solution to (B.1) for the smallest value of k. For $k=k_0=0$, we get an integer solution $x(0)=(0, 0, -24, -54, 45, 14)^T$

$$\text{and } F=\begin{bmatrix} 0 & 1 & 7 & 12 & -10 & -3 \\ 1 & 0 & -7 & -9 & 7 & 3 \end{bmatrix}^T.$$

(NOTE: Transpose notation is used to conserve space.)

STEP IV: Similarly for $k=k_1=1$, the following solution vector $x(1)$ can be found for (B.1):

$$x(1)=(0, 0, -22, -50, 42, 13)^T$$

STEP V: Construct the difference vector f:

$$f=-[x(1)-x(0)]=-(0, 0, 2, 4, -3, -1)^T$$

STEP VI: Use the modified Fourier-Motzkin elimination method to solve

$$x(0)+Fy-kf\geqq 0$$

for integer k and integer vector y. For our example, the following inequality system is solved:

$$\begin{aligned} y_1 &\geqq 0\\ y_2 &\geqq 0\\ 7y_1-7y_2+2k &\geqq 24\\ 12y_1-9y_2+4k &\geqq 54\\ -10y_1-7y_2-3k &\geqq -45\\ -3y_1+3y_2-k &\geqq -14 \end{aligned} \tag{B.3}$$

Eliminate y_1 from the inequality system (B.3) (refer to Section 4 for the steps) to obtain:

$$\begin{aligned} y_2 &\geqq 0\\ 7y_2-3k &\geqq -45\\ 21y_2-k &\geqq -75\\ -3y_2+2k &\geqq 0\\ 3y_2-k &\geqq -14\\ -k &\geqq -26\\ 3y_2 &\geqq -2 \end{aligned} \tag{B.4}$$

Use the inequalities defined by (B.4) to eliminate y_2:

$$\begin{aligned} 75-k &\geqq 0\\ 210-10k &\geqq 0\\ 173-8k &\geqq 0\\ 89-k &\geqq 0\\ 2k &\geqq 0\\ 135+5k &\geqq 0\\ 14+k &\geqq 0\\ 2+2k &\geqq 0 \end{aligned} \tag{B.5}$$

From (B.5), bounds on k can be established as

$$0\leqq k\leqq 21.$$

Set $k=0$. Substitute $k=0$ in (B.4) to get conditional bounds on y_2 as

$$0\leqq y_2\leqq 0.$$

Set $y_2=0$. Substituting $k=0$, $y_2=0$ in (B.3), the conditional bounds on y_1 is obtained as $5\leqq y_1\leqq 4$.

Since this is infeasible, the solution $k=0$, $y_2=0$ is inconsistent. So, we backtrack to the first variable whose lower bound is less than its upper bound. Hence, backtrack to variable k. Increase the lower bound on k by one; $1 \leqq k \leqq 21$. Set $k=1$ and proceed to find conditional bounds on y_2 and y_1 as before.

Continuing in a similar manner we will finally get the following solution to satisfy (B.3):

$$k^*=3,\ y_2^*=2 \text{ and } y_1^*=5.$$

It thus follows

$$x^*=x(0)+Fy^*-k^*f=[2, 5, 3, 0, 0, 2]^T$$

is a nonnegative integer solution to the integer program (P1) and the maximum value of z is,

$$(z_0-k^*)=11.$$

REFERENCES

[1] Balinski, M. L., "Integer Programming: Methods, Uses, Computation," Management Science *12*, 253–313 (1965).

[2] Balinski, M. L. and K. Spielberg, "Methods for Integer Programming: Algebraic, Combination 1, Enumerative," In J. S. Aronofsky (Ed.), *Progress in Operations Research III* (John Wiley, New York, 1969), 195–292.

[3] Biondi, E. and R. Schmid, "An Approximate Algorithm for Discrete Linear Programming," IEEE Transactions on Systems Science and Cybernetics, SSC–5, 742–745 (1969).

[4] Blankinship, W. A., "A New Version of the Euclidean Algorithm," American Math. Monthly *70*, 742–745 (1963).

[5] Bradley, G. H. and P. N. Wahi, "An Algorithm for Integer Linear Programming: A Combined Algebraic and Enumerative Approach," Report No. 29, Dept. of Administrative Sciences, Yale University (Dec. 1969).

[6] Cook, R. A., "An Algorithm for Integer Linear Programming," Doctoral Dissertation, Washington University (Jan. 1966).

[7] Cook, R. A. and L. Cooper, "An Algorithm for Integer Linear Programming," Report No. AM 65–2, Washington University (Nov. 1965).

[8] Cushing, B. E., "The Application Potential of Integer Programming," The Journal of Business *43*, 457–467 (1970).

[9] Dantzig, G. B., "On the Significance of Solving Linear Programming Problems with Some Integer Variables," Econometrica *28*, 30–44 (1960).

[10] Gomory, R. E., "Outline of a Algorithm for Integer Solutions to Linear Programming," Bulletin of the American Math. Society *64*, 275–278 (1958).

[11] Gomory, R. E., "All-Integer Programming Algorithm," Report RC–191, IBM Research Center, (Jan. 1960).

[12] Gonzalez-Zubieta, R. H., "On Some Aspects of Integer Linear Programming," Rept. No. 16, ORC, M.I.T. (June 1965).

[13] Haldi, J., "25 Integer Programming Test Problems," Paper No. 43, Graduate School of Business, Stanford University (Dec. 1964).

[14] Richmond, T. R., "Investigation of a Generalized Euclidean Procedure for Integer Linear Programs," Doctoral Dissertation, School of Industrial Engineering, Purdue University (Aug. 1971).

[15] Rubin, D. S., "Integer Solutions of Integral Linear Systems," Rept. No. 6930, Center for Math. Studies in Business and Economics, University of Chicago (Aug. 1969).

[16] Saaty, T. L., *Optimization in Integers and Related Extremal Problems* (McGraw Hill, New York, 1970).

[17] Thompson, G. L. "The Stopped Simplex Method: I. Basic Theory for Mixed Integer Programming," Revue Francaise de Recherche Operationelle *8*, 159–182 (1964).

[18] Trauth, C. A. and R. E. Woolsey, "Practical Aspects of Integer Linear Programming," Rept. No. SC–R–66–925, Sandia Corp. (Aug. 1966).

[19] Trauth, C. A. and R. E. Woolsey, "IPSC–A Machine Independent Integer Linear Program," Rept. No. SC–RR–66–433 (July 1966).

[20] Wahi, P. N., "Theory and Computation of Integer Linear Programs," Doctoral Dissertation, Yale University (Aug. 1969).

MATHEMATICAL ASPECTS OF THE $3 \times n$ JOB-SHOP SEQUENCING PROBLEM*

Wlodzimierz Szwarc

School of Business Administration
University of Wisconsin-Milwaukee
Milwaukee, Wisconsin

ABSTRACT

The paper discusses mathematical properties of the well-known Bellman-Johnson $3 \times n$ sequencing problem. Optimal rules for some special cases are developed. For the case $\min B_i \geq \max A_j$ we find an optimal sequence of the $2 \times n$ problem for machines B and C and move one item to the front of the sequence to minimize (7); when $\min B_i \geq \max C_j$ we solve a $2 \times n$ problem for machines A and B and move one item to the end of the optimal sequence so as to minimize (9).

There is also given a sufficient optimality condition for a solution obtained by Johnson's approximate method. This explains why this method so often produces an optimal solution.

1. INTRODUCTION

R. Bellman stated in 1954 the following problem (which will be called the $3 \times n$ problem). There are given items 1, 2, . . ., n each to be processed on three machines A, B, and C in the same order ABC. Given the processing times, the problem is to find an operation schedule for each machine so as to minimize the total elapsed time[1] necessary to process all the items. S. M. Johnson [3] showed that without loss of optimality one can assume the same processing order for each machine.[2]

At this point, only two special cases had been solved.

CASE 1 ([3]): $B_i \leq \max(\min_i A_i, \min_i C_i) \qquad \forall_i$

CASE 2 ([1]): $B_i \geq \min(\max_i A_i, \max_i C_i) \qquad \forall_i$

where A_i, B_i, and C_i are the processing times for item i on machines A, B, and C, respectively.

REMARK: Reference [6] solved the case

$$B_i \geq \max(\max_i A_i, \max C_i) \qquad \forall_i \quad \text{(CASE 3)}.$$

This paper deals primarily with the mathematical aspects of the problem. It presents a unified and simple justification for the solution methods to all special cases.[3] For case 2 it offers a much simpler solution procedure than Reference [1]. Section 4 answers the questions when and why Johnson's approximate method produces an optimal solution.

---

*Presented at the 41st ORSA National Meeting in New Orleans on April 28, 1972.

[1] Or, equivalently, the total idle time on the last machine, i.e., machine C.

[2] This is, however, no longer true if the number of machines exceeds three.

[3] The proof of case 1 in [3] is not complete due to the fact that the terms $H_v + K_u$, $H'_v + K'_u$ for $u = j, j+1, j+2 \leq v \leq n$ are missing in the expressions on page 66 (bottom).

One should note that the presented solution procedures (including Mitten's method for solving case 1 [4]) provides a considerable insight into the nature of the problem.

2. THE 2 × n PROBLEM

Let $p=(p_1, \ldots, p_n)$ be the processing order for the $2 \times n$ problem.
(machines A and B)

Then (see [1])

$$g(p)=\max_{1\leq u\leq n}\left(\sum_{i=1}^{u} A_{p_i}-\sum_{i=1}^{u-1} B_{p_i}\right) \tag{1}$$

denotes the total waiting time (to be minimized) of machine B. Mitten's equivalent version of Johnson's solution method ([4]) of this problem is as follows:

1) Divide the items into two disjoint groups s and s' where

$$s=\{i|A_i-B_i<0\}, \qquad s'=\{i|A_i-B_i\geq 0\}.$$

2) Place the elements of s in front of the elements of s'.
3) Arrange the items of s in a nondecreasing order of A_i.
4) Arrange the items of s' in a nonincreasing order of B_i.

This optimal $2 \times n$ rule immediately leads to the following important corollary.

COROLLARY 1: Removal or addition of some items in a $2 \times n$ problem does not affect the optimal (relative) ordering of the remaining items.[4]

3. SPECIAL CASES OF THE 3 × n PROBLEM

Let $p=(p_1, \ldots, p_n)$ be a processing order in the $3 \times n$ problem. Define

$$K_u=\sum_{i=1}^{u} A_{p_i}-\sum_{i=1}^{u-1} B_{p_i} \tag{2}$$

and

$$H_v=\sum_{i=1}^{v} B_{p_i}-\sum_{i=1}^{v-1} C_{p_i}. \tag{3}$$

Then ([1] or [6]) the total idle time of one machine C is equal

$$g(p)=\max_{1\leq u\leq v\leq n}(H_v+K_u). \tag{4}$$

CASE 1: There are two subcases: for all i and j

a) $B_i \leq A_j$

b) $B_i \leq C_j$.

---

[4] Suppose we removed items 1 and 6 from a 2×7 problem with 2137645 as an optimal solution. Then 23745 is an optimal solution of the 2×5 problem. This corollary, however, is not true for the $3 \times n$ case.

CASE 1a: implies $K_u \leqslant K_{u+1} \leqslant \ldots \leqslant K_v$ and (4) becomes

$$g(p) = \max_{1 \leqslant v \leqslant n} (H_v + K_v) = \max_{1 \leqslant v \leqslant n} \left[\sum_{i=1}^{v} (A_{p_i} + B_{p_i}) - \sum_{i=1}^{v-1} (B_{p_i} + C_{p_i}) \right]. \tag{5}$$

Consider Case 1b. Then $H_{v+1} \leqslant H_v \leqslant \ldots \leqslant H_u$ and

$$g(p) = \max_{1 \leqslant u \leqslant n} (H_u + K_u)$$

which is identical with (5).

Comparing (5) and (1) we see that case 1 has a $2 \times n$ "structure." Hence to solve the problem apply the optimal $2 \times n$ rule assuming the operating times of item i on the first and second machine to be equal $A_i + B_i$, $B_i + C_i$, respectively.

CASE 2: We have to distinguish two cases

a) $B_i \geqslant A_j$

b) $B_i \geqslant C_j$ for all i and j.

CASE 2a: Then $K_u \geqslant K_{u+1}$ and (4) becomes

$$g(p) = \max_{1 \leqslant v \leqslant n} (H_v + K_1) = A_{p_i} + \max_{1 \leqslant v \leqslant n} H_v. \tag{6}$$

We can assume without loss of generality that $p = (p_1, \ldots, p_n)$ minimizes $\max H_v$.

To find a sequence minimizing (6) one may proceed as follows: Denote by $\bar{p}$ an arbitrary permutation of numbers $1, \ldots, n$ where p_i occupies the first place. Examine for each $i = 1, \ldots, n$ an expression

$$A_{p_i} + \min_{\bar{p}} \max_{1 \leqslant v \leqslant n} H_v \tag{7}$$

and choose a permutation for which (7) assumes the smallest value.

Consider (7) for a fixed i. The problem of finding a $\bar{p}$ minimizing $\max H_v$ is equivalent to a $2 \times (n-1)$ problem where machines B and C operate items $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$ and where B and C cannot start earlier than $A_{p_i} + B_{p_i}$ and $A_{p_i} + B_{p_i} + C_{p_i}$ (units of time), respectively.

It is easy to show (see [6], pp. 137–138) that the latter problem is equivalent to a "regular" $2 \times (n-1)$ problem (without the delay restriction).

Since $p = (p_1, \ldots, p_n)$ minimizes $\max H_v$ (assumption) we conclude (by corollary 1) that sequence

$$p_1, p_2, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$$

is the optimal solution for the $2 \times (n-1)$ problem and $\bar{p} = p_i, p_1, \ldots, p_n$ minimizes the second term in (7).

Hence the solution method for case 2a is as follows:

STEP 1: Determine the optimal schedule, say, $p = (p_1, \ldots, p_n)$ for the $2 \times n$ problem (machines B and C).

STEP 2: Consider n sequences $p^{(i)}$, $i=1, \ldots, n$ where $p^{(1)}=p$ and $p^{(i)}=(p_i, p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n)$ for $i>1$. The optimal solution is the sequence minimizing (6).

REMARK 1: Since p determined in step 1 minimizes max H_v we may only examine sequences $p^{(i)}$ for which $A_{p_i} < A_{p_1}$. (see example)

Example 1

Consider the following 3×6 problem.

	A	B	C
1	6	8	4
2	8	9	9
3	5	7	12
4	7	10	5
5	4	11	8
6	3	8	10

Applying step 1 we find $p=(3\,6\,2\,5\,4\,1)$.

According to Remark 1 we have to consider two more sequences

$$p^{(4)}=5\,3\,6\,2\,4\,1 \quad \text{and} \quad p^{(2)}=6\,3\,2\,5\,4\,1$$

	B	C
3	7^{7}	12^{21}
6	8^{15}	10^{31}
2	9^{24}	9^{40}
5	11^{35}	8^{48}
4	10^{45}	5^{53}
1	8^{53}	4^{57}

(1)

	B	C
5	11^{7}	8^{19}
3	7^{18}	12^{31}
6	8^{26}	10^{41}
2	9^{37}	9^{50}
4	10^{47}	5^{55}
1	8^{53}	4^{59}

(2)

	B	C
6	8^{8}	10^{18}
3	7^{15}	12^{30}
2	9^{24}	9^{39}
5	11^{35}	8^{47}
4	10^{45}	5^{52}
1	8^{53}	4^{57}

(3)

REMARK 2: The numbers above the operation times denote the corresponding completion (or elapsed time) of processing a sequence of items on a machine.

So for instance, $t(3\,6\,2\,5, B)=35$, $t(3\,6\,2\,5, C)=48$ (Table 1)

$t(6\,3\,2\,5\,4\,1, C)=57$ (Table 3).

These numbers are found by a rule which is illustrated by the following examples (Table 3).

$$t(6\,3\,2\,5\,4\,1, C)=\max\,[t(6\,3\,2\,5\,4\,1, B), t(6\,3\,2\,5\,4, C)]+C_1=$$
$$=\max\,(53, 52)+4=57$$

$t(6, C)=t(6, B)+C_6=8+10=18$, $t(6\ 3\ 2\ 5, B)=t(6\ 3\ 2, B)+B_5=24+11=35$.

Subtracting $\sum_{i=1}^{n} C_i=48$ from the total elapsed time, we find the corresponding $\max_v H_v$. Using (6) determine

$$g(p)=A_3+57-48=5+9=14$$
$$g(p^{(4)})=A_5+59-48=4+11=15$$
$$g(p^{(2)})=A_6+57-48=3+9=12.$$

Hence $p^{(2)}=6\ 3\ 2\ 5\ 4\ 1$ is the optimal sequence.

CASE 2b: Here $H_v \leq H_{v+1}$ and (4) assumes a form

$$g(p)=\max_{1\leq u\leq n}(H_n+K_u)=\sum_{i=1}^{n} B_{p_i}-\sum_{i=1}^{n-1} C_{p_i}+\max_{1\leq u\leq n} K_u. \tag{8}$$

Introduce

$$g'(p)\overset{df}{=}g(p)-\sum_{i=1}^{n} B_{p_i}+\sum_{i=1}^{n} C_{p_i}=g(p)+\text{constants}$$

$$g'(p)=C_{p_n}+\max_{1\leq u\leq n} K_u. \tag{9}$$

The problem is to find a sequence $p_1, \ldots, p_n$ minimizing (9).[5] Reasoning the same way as in case 2a we come up with the following solution method:

STEP 1: Determine the optimal schedule, say $p=(p_1, \ldots, p_n)$ for the $2\times n$ problem (machines A and B).

STEP 2: Consider n sequences $p^{(i)}$, $i=1, \ldots, n$ where

$$p^{(i)}=(p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n, p_i) \text{ for } i<n \text{ and } p^{(n)}=p.$$

The optimal solution is the sequence minimizing (9).

REMARK 3: Actually one may only examine sequences $p^{(i)}$ for which $C_{p_i}<C_{p_n}$.

CASE 3: Observe that Case 3 ⟨=⟩ case 2*a* or case 2*b*. Hence (4) becomes

$$g(p)=H_n+K_1=\sum_{i=1}^{n} B_{p_i}-\sum_{i=1}^{n-1} C_{p_i}+A_{p_1}.$$

Introducing $g'(p)=g(p)-\sum_{i=1}^{n} B_{p_i}+\sum_{i=1}^{n} C_{p_i}$, we get

$$g'(p)=A_{p_1}+C_{p_n} \tag{10}$$

[5] This sequence minimizes (8).

The optimal rule for this case is as follows:

(1) Find

$$\min_{i\neq j}(A_{p_i}+C_{p_j})=A_{p_r}+C_{p_s} \tag{11}$$

(2) Any sequence of the form $p_r, \ldots, p_s$ is optimal.

Example 2 (from [1])

Consider the following 3×4 problem.

	A	B	C
1	5	9	6
2	8	11	5
3	7	8	2
4	4	12	4

Here we deal with case 3[6] since $\min B_i=8\geq\max(\max A_i, \max C_i)=8$.
Determine

$$\min_{i\neq j}(A_{p_i}+C_{p_j})=A_4+C_3=4+2=6$$

Hence 4 1 2 3 and 4 2 1 3 are optimal sequences.

CASE 4: $B_1=B_2=\ldots=B_n$.

PROPOSITION 1: If one can arrange the items so that $A_{p_1}\leq A_{p_2}\leq\ldots\leq A_{p_n}$ and $C_{p_1}\leq C_{p_2}\geq\ldots\geq C_{p_n}$ then $p=(p_1, \ldots, p_n)$ is the optimal sequence of the $3\times n$ problem.

PROOF: For simplicity assume $p=(1, 2, \ldots, n)$. Let $p'=(1, 2, \ldots j-1, j+1, j, j+2, \ldots, n)$. Examine

$$g(p)=\max_{1\leq u\leq v\leq n}(H_v+K'_u) \text{ and } g(p')=\max_{1\leq u\leq v\leq n}(H'_v+K'_u)$$

where (see [3] or [6])

$$K'_{ij}=K_j-A_j+A_{j+1} \qquad K'_{j+1}=K_{j+1}+B_j-B_{j+1}$$
$$H'_j=H_j-B_j+B_{j+1} \qquad H'_{j+1}=H_{j+1}+C_j-C_{j+1}$$

By assumption

$$B_j=B_{j+1}=\text{const.}, A_j\leq A_{j+1}, C_j\geq C_{j+1}. \tag{12}$$

---

[6] The authors of [1] treated this example as a $2a$ case. To solve the problem they must solve n $2\times(n-1)$ problems and find the optimal sequence according to criterion 5 on page 136.

Hence

$$K_j \leq K'_j,\ H_j \leq H'_j,\ K_{j+1} \leq K'_{j+1},\ H_{j+1} \leq H'_{j+1}$$

Obviously,

$$K_u = K'_u,\ H_v = H'_v \qquad \text{for } u, v \neq j, j+1.$$

Therefore

$$g(p') \geq g(p)$$

Since (12) is transitive to any $p^* \neq p$ there exists a sequence of intermediate permutations

$$p^*, \ldots, p$$

with $g(p^*) \geq \cdots \geq g(p)$

Hence, p is optimal, QED

4. SUFFICIENT OPTIMALITY CONDITIONS FOR JOHNSON'S APPROXIMATE METHOD

Given a $3 \times n$ problem define a corresponding $2 \times n$ time lag problem where the items are operated by machines A and C and where B_i is the waiting time for item i after it is finished on machine A and before it starts on machine B. The processing order is AC for all items. We also assume[7] that the processing order (to be found) is the same for both machines.

Note that each $3 \times n$ problem can be considered as a $2 \times n$ time lag problem. The opposite statement is not true since the time lags B_i and B_{i+1} may overlap,[8] whereas in the $3 \times n$ they cannot. (Machine B cannot operate two items simultaneously.) For simplicity assume $p = (1, \ldots, n)$. Let x_i be the idle time of machine C before it starts operating item i. Then

$$x_1 = A_1 + B_1,\ x_2 = \max\ (A_1 + A_2 + B_2 - C_1 - x_1, 0) \qquad \text{and}$$

$$x_n = \max\left(\sum_{i=1}^{n} A_i + B_n - \sum_{i=1}^{n-1} C_i - \sum_{i=1}^{n-1} x_i, 0\right). \qquad \text{Hence}$$

$$g(p) \overset{df}{=} \sum_{i=1}^{n} x_i = \max\left(\sum_{i=1}^{n} A_i - \sum_{i=1}^{n-1} C_i + B_n, \sum_{i=1}^{n-1} x_i\right) = \max_{1 \leq u \leq n}\left(\sum_{i=1}^{u} A_i - \sum_{i=1}^{u-1} C_i + B_u\right)$$

Finally,

$$g(p) = \max_{1 \leq u \leq n}\left[\sum_{i=1}^{u} (A_i + B_i) - \sum_{i=1}^{u-1} (B_i + C_i)\right] \tag{13}$$

which is identical with (5). This leads to the following corollary.

COROLLARY 2: The method for case 1 solves the $2 \times n$ time lag problem. The method for case 1, known as the Johnson method, can be used as an approximate method to any $3 \times n$ problem. A question, however, arises whether the obtained sequence is optimal. Up to now the only way to find this out was to examine all $n!$ sequences.

---

[7] This is a relevant restriction.

[8] Note that this cannot happen in case 1.

Using Corollary 2 and the fact that $2 \times n$ time lag problem is less restrictive than the corresponding $3 \times n$ problem we offer the following procedure.

STEP 1: Solve the $2 \times n$ time lag problem using the method for case 1. Let $p = (p_1, \ldots, p_n)$ be the optimal solution. Find the minimal elapsed time $\left(t(p) = g(p) + \sum_{i=1}^{n} C_i\right)$.

STEP 2: Determine $T(p)$ the total elapsed time for the $3 \times n$. The following corollary holds.

COROLLARY 3: Equality $T(p) = t(p)$ is a sufficient condition for p to be an optimal solution of the $3 \times n$ problem.[9]

REMARK 4: The above procedure establishes lower and upper bounds for the minimal elapsed time, call it T.

$$t(p) \leqslant T \leqslant T(p)$$

Example 3 ([2], p. 309, Example 2)

	A	B	C
1	9	13	6
2	7	7	20
3	6	4	8
4	8	3	10
5	20	7	2
6	10	2	13

	A+B	B+C
1	22	19
2	14	27
3	10	12
4	11	13
5	27	9
6	12	15

As seen, neither of the cases considered in section 3 holds. Applying step 1, determine $p = (3\ 4\ 6\ 2\ 1\ 5)$, and $t(p) = g(p) + \Sigma C_i = [105 - \Sigma(B_i + C_i)] + \Sigma C_i = 105 - \Sigma B_i = 105 - 36 = 69$. (Table 4) [10]

	A+B	B+C
3	10^{10}	12^{22}
4	11^{21}	13^{35}
6	12^{33}	15^{50}
2	14^{47}	27^{77}
1	22^{69}	19^{96}
5	27^{96}	9^{105}

(4)

	A	B	C
3	6^{6}	4^{10}	8^{18}
4	8^{14}	3^{17}	10^{28}
6	10^{24}	2^{26}	13^{41}
2	7^{31}	7^{38}	20^{61}
1	9^{40}	13^{53}	6^{67}
5	20^{60}	7^{67}	2^{69}

(5)

Since (Table 5) $T(p) = 69 = t(p)$ program 3 4 6 2 1 5 is an optimal solution of the problem.

[9] Another sufficient condition via some perturbation procedure was established in [5].

[10] The numbers above the operation times (tables 4 and 5) are completion times—see Remark 2.

REFERENCES

[1] Arthanari T. S. and A. C. Mukhopadhyay, "A Note on a Paper by W. Szwarc," Nav. Res. Log. Quart. *15*, 135–138 (1971).

[2] Giglio R. J. and H. M. Wagner, "Approximate Solutions to the Three-machine Scheduling Problem," Operations Research *12*, 305–324 (1964).

[3] Johnson, S. M., "Optimal Two and Three-Stage Production Schedules with Setup Times Included," Nav. Res. Log. Quart. *1*, 61–68 (1954).

[4] Mitten, L. G., "Sequencing *n* Jobs of Two Machines with Arbitrary Time Lags," Management Science, *5*, 293–298 (1959).

[5] Raghavachari, M., "On an Approximate Solution to the 3-Machine Sequencing Problem," J. Math. Sci. *4*, 47–50 (1969).

[6] Szwarc, W., "On Some Sequencing Problems," Nav. Res. Log. Quart. *15*, 127–155 (1968).

THE DEVELOPMENT AND EVALUATION OF A COST-BASED COMPOSITE SCHEDULING RULE

Sumer C. Aggarwal
Bruce A. McCarl

Pennsylvania State University

ABSTRACT

A cost-based composite scheduling rule is developed and evaluated in comparison with three other well-researched scheduling rules—SPT, S/OPN, and SST. This cost rule permits the optimization of more than one performance measure at a time. The priority number that is used for scheduling operations through each machine group is based on four separate performance measures—(1) In-process Inventory, (2) Facilities Utilization, (3) Lateness, and (4) Mean Setup Time. The factorial experimental design involved three factor levels of loads, three factor levels of cost, and three factor levels of mean time. Analysis of variance was performed on each of the five output measures to study the effects of each of the three factors on each individual rule. Rank-order comparisons between rules were also made; and, finally, general conclusions with regard to the effectiveness and flexibility of the Cost Rule were drawn.

INTRODUCTION

In the past, a large number of job shop scheduling rules were developed. Each of the rules uses some time element as the criterion by which the decision to schedule individual jobs is made. Each rule resolves the conflict whenever two or more jobs (or operations of jobs) are waiting for the services of a single machine. A rule helps to identify the job with the maximum priority (in relation to a given performance criterion) among those waiting for service at a particular machine group.

Scheduling rules previously reported [4, 5] have used performance criteria such as minimizing the waiting time of the high value jobs; maximizing the percentage utilization of men or machines; minimizing work-in-process inventory (or minimizing the mean flow time of jobs); and minimizing the average lateness of jobs. The above criteria are in no way independent; each may be highly conflicting with the other criteria. For example, if a rule has primarily been structured to minimize the average lateness of jobs, it may not maximize machine utilization and may not minimize the average work-in-process inventory. Unfortunately, there cannot be any "pure" scheduling rules, that will optimize only one performance measure at a time. Whenever the manager of a job shop tries to optimize more than one performance criterion, he must always establish a suboptimal trade-off with other criteria. In such a complex area of investigation, simulation seems to be the only effective method of comparing different scheduling rules.

It may also be noted that each of the above criteria utilizes time as the basic element for assigning priorities to jobs and thus for deciding on the job sequence. In real life, often the decisions are made using cost as a performance measure, so a question arises: Why can managers not use operations costs or some cost index for assigning priorities to individual operations?

The objective of this investigation is to develop a scheduling rule that will optimize the overall operating costs of processing jobs through a shop without adversely affecting the other measures of

performance (average lateness, average work-in-process inventory, or machine utilization). In this study the effectiveness of a cost-based rule is compared with three other well-researched rules: the SPT Rule (i.e., the shortest processing time rule), the SST Rule (i.e., the shortest set-up time rule), and the S/OPN Rule (i.e., the slack per remaining operation rule).

LOGIC AND STRUCTURE OF A COST-BASED SCHEDULING RULE

Researchers in the field of scheduling have generally stressed four major performance criteria: in-process inventory, facilities utilization, lateness, and mean setup time. Each criterion represents a different component of operational costs, and each may be translated into a cost index as explained below. These four elements of operations cost cover almost all the costs related to scheduling.

Before the individual cost indices for each of the four performance measures are explained, it must be emphasized that these indices are not the actual costs. The indices merely represent the relative magnitude of their costs and are computed for each operation remaining at the beginning of each day (assuming single shift working). The sum of these four indices constitutes the priority number of the given operation. The number does not represent dollars, dollar-hours, or any specific units; rather, it represents only the relative importance of operations (jobs) existing in a queue. On a particular day, the operation (job) with the highest priority number is scheduled first, the operation with the second highest priority number second, and so on.

(1) In-Process Inventory Cost Index

The in-process inventory cost component must take into account the dollar value of the job and the amount of time a job is required to wait in the queue. Consequently, the in-process inventory cost will primarily be proportional to the dollar value of a job and to the number of days the job stays in the queue. This component cost index may be represented by

$$C_1 V_i (D_{ijn} - T),$$

where

C_1 = The cost in dollars of carrying inventory worth one dollar for one day;

V_i = The estimated average of two dollar values of the job i — (1) the value at the time of entrance, and (2) the value after completion [a trial simulation run of 1,000 jobs indicated that updating V_i as the job moves from one operation to the next does not make much of a difference in the total of operations costs, hence it was decided to use this average dollar value];

D_{ijn} = The estimated due date of the jth operation of the ith job in the nth machine group (noting that no job with more than 30 slack days is released to the shop, and hence it is the dollar value of the job that mostly affects the priority);

T = The current day at the beginning of which the priority numbers are calculated for all remaining operations and for the new jobs that arrived during the previous shift.

As soon as a job becomes late, the inventory cost index becomes negative; but, actually a negative inventory cost is considered as a part of the Lateness Cost Index. (The reasons for such a merger are explained below under the section "Lateness Cost Index.")

(2) Facilities Utilization Cost Index

It is well known that when a job shop uses the SPT rule for sequencing individual jobs through its machine centers, then its utilization of facilities improves [3]. This, in turn, reduces the mean hourly

cost of machine utilization. Therefore, this component index may be represented by a number inversely proportional to the processing time of the given operation. Further, average processing time varies from one machine group to another and, therefore, an operation may be weighted in direct proportion to the mean processing time of its respective machine group. Hence, this component cost index may be expressed by

$$C_{2n}\frac{K_{1n}}{P_{ijn}},$$

where

C_{2n} = Average hourly cost of machine utilization for the nth machine group (dollars),
K_{1n} = The mean processing time per operation for the nth machine group (hours),
P_{ijn} = Estimated processing time for the jth operation of the ith job in the nth machine group (hours).

As P_{ijn} of the above term approaches zero, this index value tends to approach infinity; and, in consequence, the overall priority number for the operation becomes extremely high, which means quick dispatch of the operation. In case $P_{ijn}=o$, which means no such operation exists, and hence no value of this index need be calculated. An alternative to this cost-index can be based on "look ahead approach"—give priority to the operation that next goes to the shortest queue or to the operation that next goes to the costliest machine group—but past studies indicate that the improvement in performance as a result of these approaches is not enough to justify the additional expense on the required computer time [3, 4].

(3) Lateness Cost Index

The penalty for late completion of individual jobs represents opportunity losses due to locked-up capital. Late deliveries cause a loss of future orders. As the average amount of lateness increases, the probability of obtaining repeat orders decreases rapidly [8]. Therefore, it seems reasonable to assume that the lateness cost index of an operation increases in direct proportion to the square of the number of days a job is late, and may be represented by the term

$$C_{3i}\cdot\Delta\cdot(T-D_{ijn})^2,$$

where

C_{3i} = cost of lateness of the ith job per day,
= $2C_1V_i$ (i.e., we assume that the lateness cost rate is two times the in-process inventory holding rate),
$\Delta\cdot(T-D_{ijn})$ = ith job lateness in number of days, taking into account that
$\Delta=1$, when $T>D_{ijn}$, and
$\Delta=0$, otherwise.

With increase in lateness, the square term of the above expression makes the lateness cost index extremely high. However, it must be noted that when the lateness is of one, two or three days only, then the severe lateness penalty caused by the square term is reduced by an amount equal to the negative inventory cost component mentioned earlier (because for lateness these two components are taken together). In fact, when $\Delta=1$, the lateness cost index is given by

$$C_1V_i(D_{ijn}-T)+C_{3i}\cdot\Delta\cdot(T-D_{ijn})^2.$$

(4) Mean Setup Time Cost Index

Wilbrecht and Prescott have established that giving priority to the job with the shortest actual setup time (which is in no way sequence dependent), improves overall job-shop performance [9]. They clearly state that there is no easy explanation of SST's performance because so many of the variables are interacting simultaneously. They also state that the SST Rule definitely reduces mean waiting time for the machines as well as mean waiting time for the jobs. The improved performance given by the SST Rule may be interpreted from a slightly different angle. When the mean of the actual setup times increases, the overall mean hourly setup cost decreases. So once again, just as in the facilities utilization cost index, the mean setup time cost index of a given operation will be inversely proportional to its setup time and directly proportional to the mean setup time of its respective machine group. The mean setup time cost index can be expressed by the term

$$C_{4n} \cdot \frac{K_{2n}}{S_{ijn}},$$

where

C_{4n} = Mean hourly setup time cost for the nth machine group (dollars),

K_{2n} = Mean setup time per operation for the nth machine group (hours),

S_{ijn} = Estimated setup time for the jth operation of the ith job on the nth machine group (hours).

As S_{ijn} approaches zero, this cost index approaches infinity and, when $S_{ijn}=0$ (which means there is no such operation), no value of this index need be calculated. This means that a very small value of S_{ijn} will make the value of this component cost index for a given operation very large, and as a result that operation will be quickly dispatched.

The Composite Priority Rule

Each of the above four cost indices was primarily designed to reflect the costs of its respective components as realistically as possible. In the past, it has been shown that the composite priority rules, having few constituent priority indices, were able to achieve multiple performance objectives [6, 7]. Therefore, it seems quite appropriate to combine the four cost indices into one single composite expression, and call it the "Cost Rule," which is

$$Z_{ijn}=C_1V_i(D_{ijn}-T)+C_{2n}\frac{K_{1n}}{P_{ijn}}+C_{3n}\cdot\Delta\cdot(T-D_{ijn})^2+C_{4n}\frac{K_{2n}}{S_{ijn}}.$$

Here,

Z_{ijn} = The priority number (i.e., the total cost index) for the ith operation of the jth job on the nth machine group. (Note Z_{ijn} is only the priority number and not the total cost of the operation. The equation for calculating cost of completion of a job is given later)

At the start of every day, assuming a single working shift, priority numbers must be calculated for each of the remaining job operations in the shop (including the operations of the jobs newly arrived during the previous shift). During a particular day, as soon as an operation of a job is completed, the job immediately moves to the queue of another machine group, required by its next operation. In this way, each job progresses through the various machine groups until it is completed.

TABLE I. *Parameter Values of the Different Machine Groups**

Item	Machine Groups					
	1	2	3	4	5	6
Number of Machines in the Groups	3	2	2	1	1	1
Mean Setup Time (hr)	0.4	0.5	0.3	0.6	0.2	0.1
Mean Setup Cost ($/hr)	10	12	9	14	6	5
Mean Processing Time (hr)	2.5	1.5	1.0	0.5	1.0	0.5
Mean Processing Cost ($/hr)	7	10	6	10	5	4
Mean Setup Time plus Processing Time (hr)	2.9	2.0	1.3	1.1	1.2	0.6
Mean Service Rate per hour	1.03	1.00	1.53	0.91	0.83	1.67
The Probability, with which the Operation Selects the Group	0.148	0.143	0.219	0.130	0.119	0.241

*These values were taken from the past records of a job-shop.

Simulation Model and Assumptions

The computer language chosen for this analysis was SIMSCRIPT II, using the SHARE compiler with an IBM 360/67 computer. Simulation experiments were conducted for a shop that had six machine groups. Different machine groups had different numbers of equivalent machines (see Table I). By equivalent machines, we mean that, within a particular machine group, all its machines can accomplish the same type of operations with equal efficiency. Any job can be allotted to any of the machines in the group, as a machine becomes available. This small number of machines and machine groups was selected because previous studies suggested that six machines were adequate to produce "Job-Shopiness" [1, 9].

The generation program for the jobs generated on a daily basis (assuming one working shift). The job arrivals followed a Poisson process. The program also generated a sequence of machine group destinations (operations) for each job, and permitted completely random movement of jobs from one group to another. There could be from one to six operations in a particular job, and the number was uniformly distributed. The probability of an operation selecting a particular machine group was in accordance with the mean service rate of the given machine group, so that the load on each individual machine group remained in proportion to its installed capacity.

The setup-time distribution and the process-time distribution of each machine group was assumed to be of negative exponential form. The parameter values of these distributions are given in Table I. With these distributions, the setup time and operation time of each individual operation were also generated. The mean value of each job was randomly selected from a uniform distribution between $1 and $100. Due dates for the jobs were determined randomly from a uniform distribution, ranging from 1 to 30 days. The uniform distributions were selected to represent a pure job shop with a large product

mix and large variations in dollar values of the jobs. Arbitrary due dates of the jobs are common, because often the managers of job shops cannot force delivery dates on customers. Due dates for individual operations were calculated backwards, starting from the given due date of the job, using the expression

Slack time allowed after completion of $(j-1)$st operation of job i = (Due-date for job $i-j$th operation due-date)

$$= \frac{5}{8} \sum_{j}^{j=J} \sum_{n}^{n=N} (S_{ijn} + P_{ijn}),$$

where J is the last operation of ith job, and N is the last machine group.

In the above expression division by eight converts hours into days (shifts). Multiplication by five (choice of five is explained below) means that the allowed slack time for a particular operation must be five times the time actually required. Five are chosen after several simulation trials. The trial with three imposed heavy lateness penalties, whereas the trial with seven resulted in a large number of early completed jobs and high in-process inventories. The slack of five times was chosen as a compromise between the two extreme outcomes. This multiplier was found to hold for all the six machine centers. Individual operation due dates were rounded off to integer values.

The simulation was also programmed to calculate the cost of completion of each individual job. The simulation used the actual costs of setup and processing times for the operation; the actual due date of the job; and the given dollar value of the job. The expression used for calculating the completion cost of each job was:

$$\text{Cost of a completed job} = C_1 V_i (D_i - T) + \sum_j C_{2n} \cdot P_{ijn} + C_{3n} \cdot \Delta \cdot (T - D_i)^2 + \sum_j C_{4n} \cdot S_{ijn}.$$

First term on the right hand side of the above expression represents undesirable inventory carrying cost; the second and fourth terms jointly make up the machine utilization cost; and the third term stands for the lateness cost. There is no direct way to allocate the idle time cost to individual operations; whereas in actual practice the idleness cost is always recovered by way of estimated machine hourly rates that do take into account idleness of men and machines. Hence the above expression need not have a separate term for the idleness cost.

Our model allowed jobs to be processed by a particular machine group more than once, as is common is real job-shop situations. However, the model did not permit preemption, flexibility in routing of operations, or machine breakdowns.

EXPERIMENTAL DESIGN

The objective of this study was to compare the composite cost-based rule with three other well-known scheduling rules on five major criteria of performance. Symbols used have the following meanings:

1) The average total cost per job $= TPC$,
2) The percentage of jobs completed late $= PJL$,

3) The percentage of idle-time $= PIT$,
4) Percentage of jobs waiting in queues, (these also represent the in-process inventories) $= PJIQ$,
5) Percentage of late-jobs waiting in queues $= PLJIQ$.

The four scheduling rules that were compared were:

1) Composite cost-based rule (or Cost rule) = CR,
2) Shortest processing time rule = SPT,
3) Shortest setup time rule = SST,
4) Slack per remaining operation rule = S/OPN.

It is well known that the shop load greatly influences the comparative performance of priority rules. To investigate this effect, the simulation experiments were run at three different load levels (arrival rates of jobs). Using the information given in Table I, we found that our job shop could theoretically process a mean number of 15.8 jobs per day at 100 percent utilization. Therefore, we selected three load conditions (i.e., mean arrival rates of jobs):

a) Mean arrival rate of 10 jobs per day, representing *light load* conditions (i.e., 63 percent of the installed capacity) $= L_1$,
b) Mean arrival rate of 14 jobs per day, representing *medium load* conditions (i.e., 89 percent of the installed capacity) $= L_2$,
c) Mean arrival rate of 18 jobs per day, representing *heavy load* (i.e., 114 percent of the installed capacity) $= L_3$.

To study the effect of changes in hourly cost rates of machine-group setup times and machine-group processing times, we ran simulation experiments for three combinations of costs:

a) When group-wise average hourly processing-time cost and average hourly setup time cost were as shown in Table I (See Table II also) $= C_1$,
b) When group-wise processing time hourly costs were two times their respective average rates, and the group-wise setup time hourly costs were half of their respective average rates (Table II) $= C_2$,
c) When group-wise mean processing times were half of their respective average rates, and the group-wise setup time hourly costs were two times their respective average rates (Table II) $= C_3$.

Presuming that changes in the group-wise mean setup and mean processing times may affect the performance criteria results, we also tried three different combinations of group-wise mean timings:

a) When group-wise average setup times and average processing times were as shown in Table I (see Table III also) $= T_1$,
b) When group-wise mean processing times were two times their respective average processing times,

TABLE II. *Cost Rates for the Three Cost Combinations*

Cost Combination	Item	Machine Groups					
		1	2	3	4	5	6
C_1	Hourly Processing Time Cost ($)	7.0	10.0	6.0	10.0	5.0	4.0
	Hourly Setup Time Cost ($)	10.0	12.0	9.0	14.0	6.0	5.0
C_2	Hourly Processing Time Cost ($)	14.0	20.0	12.0	20.0	10.0	8.0
	Hourly Setup Time Cost ($)	5.0	6.0	4.5	7.0	3.0	2.5
C_3	Hourly Processing Time Cost ($)	3.5	5.0	3.0	5.0	2.5	2.0
	Hourly Setup Time Cost ($)	20.0	24.0	18.0	28.0	12.0	10.0

TABLE III. *Mean Times for the Three Time Combinations*

Time Combination	Item	Machine Groups					
		1	2	3	4	5	6
T_1	Mean Processing Time (hr)	2.5	1.5	1.0	0.5	1.0	0.5
	Mean Setup Time (hr)	0.4	0.5	0.3	0.6	0.2	0.1
T_2	Mean Processing Time (hr)	5.0	3.0	2.0	1.0	2.0	1.0
	Mean Setup Time (hr)	0.2	0.25	0.15	0.3	0.1	0.05
T_3	Mean Processing Time (hr)	1.25	0.75	0.5	0.25	0.5	0.25
	Mean Setup Time (hr)	0.8	1.0	0.6	1.2	0.4	0.2

and the group-wise mean setup times were half of their respective average setup times (Table III) $= T_2$,

c) When group-wise mean processing times were half of their respective average processing times, and the group-wise mean setup times were two times their respective average setup times (Table III) $= T_3$.

A careful reader will note that the balance between the load and production capacity is completely disturbed when the mean processing and mean setup times of the machine groups are changed and no corresponding adjustments are made in their installed capacities. Under changed conditions, the probabilities by which machine groups must be selected for the jobs then need to be changed. For meaningful comparisons between the results of the different time combinations, we must compute the corresponding new arrival rates and the probabilities of jobs selecting a particular machine group.

These new values were so computed that the new loads on an individual machine group for combinations T_2 and T_3 imposed exactly the same percentages of loads as in case of combination T_1. Such values have been calculated from the data given in Table I, and are recorded in Table IV.

Thus we have four scheduling rules and five performance measures. For each of these rule-measure combinations, we have a $3 \times 3 \times 3$ factorial experimental design—with three factor levels of loads, three factor levels of cost combinations, and three factor levels of time combinations.

TABLE IV. *Job Arrival Rates and the Corresponding Probabilities of Jobs Selecting a Machine Group*

Time Combination	Mean Arrival Rates that Correspond to Each Other			Corresponding Probabilities of the Jobs Selecting the Machine Groups					
	Light Load	Medium Load	Heavy Load	1	2	3.	4	5	6
T_1	10.00	14.00	18.00	0.148	0.143	0.219	0.130	0.119	0.241
T_2	5.90	8.26	10.62	0.133	0.142	0.216	0.178	0.110	0.221
T_3	11.30	15.83	20.35	0.173	0.135	0.216	0.082	0.131	0.263

STATISTICAL ANALYSIS

For each of the 27 possible combinations of factor levels, data on 1,000 jobs were collected for each of the four scheduling rules. Trial runs indicated that with light load or medium load, steady-state conditions were reached after 100 jobs, but with heavy load, the queue lengths increased. This finding is supported by the findings of Bhatt and Mann [2]. Therefore we consider that output data on jobs numbering 101–1,100 (on the last 1,000 jobs) would provide a good measure of performance.

TABLE V. ANOVA *Results on Total Cost per Job (TPC)*
(level of significance, 99%)

Source Factor	Percentage Variance for the Rule			
	C.R.	*SPT*	*SST*	*S/OPN*
Time-Combination	28.234	0	28.651	27.043
Cost-Combination	25.039	0	25.427	25.683
Load	0.0313	0	0.0764	0.326
Interaction of Time & Cost	46.601	0	45.546	46.886
Interaction of Cost & Load	0	0	0	0
Interaction of Time & Load	0.0736	0	0.234	0
Error	0.0186	100	0.625	0.0601
Grand Mean	50.888	51.783	51.595	52.546
Standard Deviation	29.943	57.773	31.053	30.666

Taking advantage of our factorial experimental design, we used the analysis of variance technique to compute the best estimates of within-cell variance for each of the performance measures for each of the scheduling rules. The results of ANOVA computation runs at the 99 percent level of significance are given in Tables V through IX. Each table contains all the results for one particular measure of performance only. The numbers entered in the columns of each table represent the percentages of variance caused by the source factors listed in the stub column. These percentages are extremely useful in making

TABLE VI. ANOVA *Results on Percentage of Late Jobs (PJL)*
(level of significance, 99%)

Source Factor	Percentage Variance for the Rule			
	C.R.	SPT	SST	S/OPN
Time-Combination	20.490	33.987	32.904	25.564
Cost-Combination	0	0	0	0
Load	66.397	44.961	45.291	17.994
Interaction of Time & Cost	0	0	0	0
Interaction of Cost & Load	0	0	0	0
Interaction of Time & Load	12.226	21.050	21.804	56.441
Error	0.854	0.00002	0.00001	0.00001
Grand Mean	5.905	3.833	5.1000	4.644
Standard Deviation	3.954	2.585	4.547	12.266

TABLE VII. *ANOVA Results on Percentage of Idle Time (PIT)*
(level of significance, 99%)

Source Factor	Percentage Variance for the Rule			
	C.R.	SPT	SST	S/OPN
Time-Combination	0.627	0.713	0.654	0.702
Cost-Combination	0	0	0	0
Load	97.79	97.967	98.057	98.354
Interaction of Time & Cost	0	0	0	0
Interaction of Cost & Load	0	0	0	0
Interaction of Time & Load	1.559	1.319	1.288	0.942
Error	0.211	0	0	0
Grand Mean	21.470	21.288	21.511	21.144
Standard Deviation	18.994	19.333	19.060	19.390

TABLE VIII. *ANOVA Results on Percentage Jobs Left in the Queues (PJIQ)*
(level of significance, 99%)

Source Factor	Percentage Variance for the Rule			
	C.R.	SPT	SST	S/OPN
Time-Combination	7.224	4.957	11.683	6.306
Cost-Combination	0	0	0	0
Load	65.776	71.190	61.616	72.496
Interaction of Time & Cost	0	0	0	0
Interaction of Cost & Load	0	0	0	0
Interaction of Time & Load	26.776	23.852	26.699	21.088
Error	0.222	0.00002	0.00001	0.109
Grand Mean	4.177	3.484	4.888	6.042
Standard Deviation	6.167	4.803	7.774	8.198

TABLE IX. *ANOVA Results on Percentage of Late-Jobs Hidden in the Queues (PLJIQ)*
(level of significance, 99%)

Source Factor	Percentage Variance for the Rule			
	C.R.	SPT	SST	S/OPN
Time-Combination	6.102	7.591	9.173	20.000
Cost-Combination	0	0	0	0
Load	71.573	46.964	66.285	20.000
Interaction of Time & Cost	0	0	0	0
Interaction of Cost & Load	0	0	0	0
Interaction of Time & Load	21.549	45.444	24.541	60.000
Error	0.811	0.00001	0	0
Grand Mean	26.949	16.611	21.888	2.855
Standard Deviation	34.893	29.512	28.944	11.059

comparisons of the effectiveness of various scheduling rules. For that purpose, the percentage variances for each rule have been placed side by side in the tables. Note that the percentages in each column add up to 100. The grand mean and the standard deviation of the means (i.e., the standard error) for each of the rules have also been entered in the bottom two rows of each table, showing the magnitude of the variance between different simulation runs.

RESULTS AND DISCUSSIONS

The results of all the 27 runs for each measure of performance were tabulated, and the rank ordering of each of the rules was determined by comparison. First rank was assigned a weight of 4; the second rank, a weight of 3; the third rank, a weight of 2; and the fourth, a weight of 1. Table X contains the details about the rank ordering of individual rules. This table also includes the total rank-ordering weights for each of the rules.

For making comparisons between the relative effectiveness of the various rules, we also prepared separate graphs for each measure of performance. Each graph had four curves—one for each rule.

TABLE X. *The Rank Ordering and the Weights of Rules on Various Measurement Criteria*

Criterion	Rule	The Number of Times the Rule Ranked				Total Rank-Ordering Weight of the Rule
		First	Second	Third	Fourth	
Total Cost of Job (TPC)	C.R.	10	14	0	3	85
	SPT	11	9	4	3	82
	SST	6	4	14	3	67
	S/OPN	0	0	6	21	33
Percent of Late Jobs (PJL)	C.R.	0	3	4	20	37
	SPT	3	21	3	0	81
	SST	0	6	20	1	59
	S/OPN	24	0	0	3	99
Percent of Idle Time (PIT)	C.R.	6	9	3	9	66
	SPT	7	12	7	1	79
	SST	4	3	15	5	60
	S/OPN	15	8	4	0	92
Percent of Jobs Left in Queues (PJIQ)	C.R.	5	18	4	0	82
	SPT	14	9	4	0	91
	SST	9	5	7	6	71
	S/OPN	6	4	8	9	61
Percent of Late Jobs Hidden in the Queues (PLJIQ)	C.R.	9	2	3	13	61
	SPT	12	12	3	0	90
	SST	9	4	13	1	75
	S/OPN	27	0	0	0	108

The abscissa on each graph represented the load factor, and the ordinate stood for one of the measures of performance. The general deductions from these graphs are also discussed below.

Total Cost Per Job (TPC): On the average the total cost per job varied between $40 and $120, with a comparatively large variance. Looking at the grand mean values of TPC (Table V), and the weights of the TPC rank orderings (Table X), it is evident that the Cost Rule performs the best. Even the rank-ordering weight of the Cost Rule is slightly better than the SPT rule. The graphs of the TPC indicated that the total cost per job does not vary by any significant amount with the increase in load. On the other hand, the effects of various time combinations (T_1, T_2, T_3) and cost combinations (C_1, C_2, C_3) are very large, as can be expected. The ANOVA results also confirm that the cost per job is mainly affected by cost, time, and their interaction only. The standard deviation for the SPT Rule (Table V) was the largest ($57.77), with the error factor constituting 100 percent of the variance. That means that the average cost per job cannot be controlled with the SPT Rule.

Percentage of Late Jobs (PJL): In general, for the time combinations T_1 and T_3, the percentage of late jobs varied between 2 and 7 percent, but for time combination T_2, this percentage went up to 30. On the rank-ordering basis (Table X), the S/OPN Rule performed best, but it had the highest standard deviation, 12.26 days (Table VI). Graphs for PJL reconfirmed that in general S/OPN was the most powerful rule, except under heavy load conditions. On PJL, the cost rule seemed to perform comparatively poorly. But in most cases the cost rule held the percentage of late jobs between 2 and 7 as compared to 2 and 3 for the S/OPN Rule. The ANOVA results (Table VI), indicate that the load factor has the largest effect, the time factor is next in importance, and load time interaction also exerts a little influence on this performance measure. If at some point of time the percentage of late jobs becomes large, even then we can use the cost rule simply by attaching greater weight to the lateness cost index.

Percentage of Idle Time (PIT): With light load, idleness in all the cases was about 42 percent; with medium load it varied from 15 to 21 percent; but with heavy load it remained at about 4 or 5 percent. It was quite evident from ANOVA results (Table VII) that, as regards PIT, all the rules were equally effective. The load factor was most prominent and caused about 98 percent of the variance for all the rules. Rank orderings of different rules (Table X) seem to give a false impression that S/OPN was superior to the other rules; but PIT graphs prepared by the authors and ANOVA results (Table VIII) clearly indicate that the rules did not make any difference whatsoever in the percentage of idleness.

Percentage of Jobs Left in Queues (PJIQ): With light load, no jobs were left in the queues under any of the rules. With medium load, the jobs left varied from 2 to 4 percent; and with heavy load, from 3 to 18 percent. This criterion may be considered as an indirect measure of in-process idle inventory in the shop. The graphs also confirmed that with light load all the rules provide exactly equivalent results. The S/OPN Rule performed worst with medium and heavy loads. With load level L_3 (heavy load), the SPT Rule and the cost rule had the least number of jobs left in the queues. On a rank-ordering basis (Table X), SPT was found to be the best, the Cost Rule the second best, and the S/OPN Rule the worst. ANOVA results (Table VIII) confirmed the relative superiority of the SPT and cost rules over the S/OPN Rule. Inspection of our graphs and ANOVA results again showed that the load was the major source factor for PJIQ; the time factor contributed slightly, but the load-time interaction effect was quite substantial.

Percentage of Late Jobs Waiting in the Queues (PLJIQ): This criterion measured the number of jobs that had already become late and were still waiting in the queues. Just like PJL, on the PLJIQ

criterion (Table IX), the S/OPN Rule was superior to all other rules. In fact, S/OPN had a grand mean of only 0.0028 percent hidden late jobs in the queues. ANOVA results (Table IX) indicate that the load factor is the main contributor to PLJIQ, and time plus time-load interaction source factors also add a little to PLJIQ. Judging from the rank ordering (Table X), one must note that S/OPN is the best performer, and the cost rule is the worst. If a situation arises where PLJIQ becomes large, that is an indication of extra heavy load, and the management must immediately audit the existing capacities of the individual machine groups.

THE VALIDITY OF RESULTS

The validity of our model can be justified in two ways. First, the SPT, SST, and S/OPN rules got the same ranking on most of the performance measures, as has been reported by various researchers in the past [3, 4]. Second, our PIT graphs clearly indicated that the actual percentage utilization for the various load levels was reasonable, and was a little less than the corresponding theoretical values, because of "job-shopiness" effects.

CONCLUSIONS

The basic objective of this study was to develop and evaluate a cost-based composite priority rule. The evaluation was carried out in comparison with the SPT, SST, and S/OPN rules. Based on the above discussions, we can draw the following general conclusions.

1. The cost rule performs best on the Total Cost Per Job (TPC) criterion and has the least Percentage of Idle Time (PIT). It is second to SPT on the in-process inventory criterion (PJIQ) and can be easily made sensitive to PJL and PLJIQ. This way, the cost rule can be very useful for multiobjective situations.

2. The S/OPN Rule is most powerful on PJL and PLJIQ only.

3. The Percentage of Idle Time (PIT) does not vary with different scheduling rules. It varies directly with the load level only; and so is in agreement with the results of the previous investigations [4, 5].

4. None of the rules has any significant effect on the Percentage of Jobs Left in Queues (PJIQ), except under heavy load conditions. On the criterion PJIQ, the SPT Rule is the best performer, and the cost rule is second.

5. The Cost Rule is the only rule which seems to have the capacity and flexibility to satisfy most of the management objectives simultaneously. If management decides to change the major emphasis from one objective to another, it can be done simply by assigning greater weight to the corresponding component cost index of the cost rule.

6. One overriding conclusion: The critical problem is to keep the productive capacity of various machine groups geared to existing shop loads. With light load, rules do not have a significant impact on any of the performance measures; with medium load, different rules perform somewhat differently from each other; but with heavy load, different rules show widely different results on each of the performance measures. Constant updating of the shop capacity can be very, very costly. That is why, for the uncertain and dynamic load conditions of a job shop and for constantly varying product mixes, the use of the Cost Rule seems to be most desirable.

NOTE: A simple FORTRAN program for this Cost Rule has been written and tried successfully in a practical situation.

REFERENCES

[1] Baker, C. T. and B. P. Dzielinsky, "Simulation of a Simplified Job-Shop Process," *IBM Systems Memorandum*, IBM Research Center, Yorktown Heights, New York (Aug. 1958).

[2] Bhatt, V. N. and S. H. Mann, "A Numerical Method for Constructing Tables for the Behavior of Some Infinite Queuing Systems," *Operations Research*, Technical Memorandum No. 82, Case Institute of Technology (1967).

[3] Conway, R. W., "An Experimental Investigation of Priority Assignment in a Job Shop," *Rand Corporation Memorandum*, RM–3789–PR (Feb. 1964).

[4] Conway, R. W., W. L. Maxwell, and L. W. Miller, *Theory of Scheduling* (Addison-Wesley Publishing Co., Reading Massachusetts, 1967).

[5] Moore, J. M. and R. C. Wilson, "A Review of Simulation Research in Job Shop Scheduling," Production and Inventory Management, Vol. 8, No. 1 (Jan. 1967).

[6] Oldziey, J. W., "Dispatching Rules and Job-Tardiness in a Simulation Job Shop," Master's Thesis, Cornell University (Feb. 1966).

[7] Orkin, C. L., "An Experimental Investigation for Shop-Loading for Setting Operations Due Dates," Master's Thesis, Cornell University (Feb. 1966).

[8] Schwartz, B. L., "A New Approach to Stockout Penalties," Management Science (Aug. 1966).

[9] Wilbrecht, J. K. and W. B. Prescott, "The Influence of Setup Time on Job Shop Performance," Management Science (Dec. 1969).

SEQUENCING WITH DUE-DATES AND EARLY START TIMES TO MINIMIZE MAXIMUM TARDINESS

Kenneth R. Baker

North Carolina State University
Raleigh, North Carolina

and

Zaw-Sing Su

University of Michigan
Ann Arbor, Michigan

1. INTRODUCTION

A fundamental problem in scheduling involves the sequencing of n jobs at a single facility. Most frequently, this problem has been examined in its static form, that is, under the assumption that all jobs are simultaneously available for processing. There are situations, however, where it is more appropriate to consider a dynamic model in which jobs become available over time. One broad area of application for dynamic models is project scheduling, in which jobs competing for a bottleneck resource are characterized by distinct early start times due to precedence relations involving other project activities. Another context for this sort of problem is the daily scheduling of computer jobs in a business data processing center, where various programs cannot be run until a data collection activity has been completed. Although this paper addresses a specific scheduling criterion, our general purpose is to illustrate how basic properties of the static problem can effectively be utilized in solving the dynamic problem.

We consider a sequencing problem consisting of n jobs and a single machine. Job j is described by the following characteristics:

a *ready time* (r_j): the earliest point in time at which processing may begin on job j (i.e., an early start time).

a *processing time* (p_j): the interval over which job j will occupy the machine, including setup.

a *due-date* (d_j): the completion deadline for job j.

The three characteristics r_j, p_j, and d_j are fixed and known in advance and no preemption is allowed in the processing of the jobs. In other words, once processing on a job is begun it may not be interrupted.

As a result of scheduling decisions, job j will be completed at time C_j and will be tardy if $C_j > d_j$. The *tardiness* of job j (T_j) is defined by

$$T_j = \max \{0, C_j - d_j\}.$$

The scheduling objective is to minimize the maximum job tardiness, which is simply

$$T_{\max} = \max_j \{T_j\}.$$

The static version of this problem, in which all r_j are equal, has a straightforward solution. Let $[j]$

denote the jth job in sequence. Then the optimal sequence in the static case is given by "Jackson's Theorem" [4]:

THEOREM 1: For the static version of the n-job single-machine problem, $T_{\max}$ is minimized by the sequence

$$d_{[1]} \leq d_{[2]} \leq \ldots \leq d_{[n]},$$

that is, by processing the jobs in nondecreasing order of their due-dates. In the dynamic version of the problem, Jackson's Theorem can again be applied if the jobs can be processed in a preempt-resume mode. With preemption allowed, sequencing decisions must be considered both at job completion times and at job ready times. The adaptation of Jackson's Theorem is then as follows:

1. At each job completion the job with minimum d_j among available jobs is selected to begin processing.
2. At each ready time, r_j, the due-date of the newly-available job j is compared to the due-date of the job being processed. If d_j is lower, job j immediately preempts the job being processed, otherwise job j is simply added to the list of available jobs.

We note that the solution to the preemptive case is easy to construct because the mechanism is a dispatching procedure, which is to say that scheduling decisions can be made at chronologically ordered points in time. If it should happen that the optimal schedule for the preemptive case does not involve preemption, then the same schedule will be optimal for the associated nonpreemptive problem. Otherwise, since all nonpreemptive schedules are contained in the set of all preemptive schedules, the optimal value of $T_{\max}$ in the preemptive case is at least a lower bound on the optimal $T_{\max}$ for the nonpreemptive schedules. This principle is the basis for our solution algorithm.

In the nonpreemptive problem, there is a sequence corresponding to each permutation of the integers $1, 2, \ldots, n$. Thus there are at most $n!$ sequences to be examined in the search for an optimum. In reality, several of these sequences need not be considered. Suppose that $r_i + p_i < r_j$. Then any schedule in which job i followed job j and which left the machine idle over the interval $(r_j - p_i, r_j)$ could be improved just by inserting job i into the idle interval. Using the terminology of job-shop scheduling [2], this means that it is sufficient to consider only *active* schedules in the search for an optimum. The number of active schedules depends on the data in a given problem, but will generally be less than $n!$. In the next section we describe a branch and bound algorithm which is based on a branching scheme which systematically enumerates all $n!$ permutations and a bounding scheme (using the preemptive adaptation of Jackson's Theorem) which allows for implicit enumeration.

2. A BRANCH AND BOUND SOLUTION ALGORITHM

The branching tree at the heart of the algorithm is essentially a tree of partial sequences, as described in [1]. Each node in the tree at level k corresponds to a partial permutation containing k jobs. Associated with each node is a lower bound on the value of the maximum tardiness which could be achieved in any completion of the corresponding partial sequence. This bound is computed by taking the schedule corresponding to the partial sequence and completing it using the preemptive adaptation of Jackson's Theorem, described in the previous section. No nonpreemptive completion of the given partial squence can attain a value of $T_{\max}$ which is lower than this bound, and the bound will frequently be quite close to the best attainable value.

The calculation of lower bounds allows the algorithm to enumerate many sequences only implicitly. If a complete sequence has been found with a value of $T_{\max}$ less than or equal to the bound associated

with some partial sequence, then it is not necessary to complete the partial sequence in the search for an optimum solution.

The branch and bound algorithm maintains a list of nodes ranked in nondecreasing order of their lower bounds. At each stage the node at the top of the list (i.e., the node corresponding to the partial sequence with the minimum lower bound) is removed and replaced on the list by several nodes corresponding to augmented partial sequences. These are formed by appending one unscheduled job to the removed partial sequence. The algorithm terminates when the node at the top of the list corresponds to a complete sequence. At this point, the complete sequence attains a value of $T_{\max}$ which is less than or equal to the lower bound associated with every partial sequence remaining on the list, and the complete sequence is therefore optimal.

In addition to the bounding technique, two other mechanisms can be used in order to reduce the computational requirements of the algorithm. First, suppose that the jobs already scheduled in a partial sequence occupy the machine beyond the maximum ready time in the job set. This means that the unsequenced jobs are, in effect, simultaneously available for the purpose of completing the sequence, and an optimal way to schedule them is given by Jackson's Theorem. When the algorithm encounters this condition it replaces the partial sequence with a completion which is constructed using Theorem 1.

A second mechanism follows from the observation that only active schedules need be enumerated. When a partial sequence containing k jobs is removed from the list, it is normally replaced by $(n-k)$ augmented sequences, but some of these can be eliminated if they do not form active partial schedules. More specifically, suppose an augmented partial sequence is to be formed by appending job j to an existing partial sequence which contains k jobs. Let $C_{\min}$ denote the minimum potential completion time which could be attained for one of the other $(n-k-1)$ unsequenced jobs if it were to replace job j in the sequence. Then if $C_{\min} < r_j$ the augmented partial sequence involving job j can be discarded immediately because it is not an active partial schedule.

The substitution and elimination mechanisms are effective in reducing the computational effort required by the algorithm, especially when n is relatively large. In addition, an upper bound calculation allows the list of nodes to be kept as small as possible. Before the tree search begins, the algorithm uses a heuristic initial phase to obtain a feasible solution to the problem. (The details of this phase will be discussed in the next section.) This initial feasible solution allows the tree search to begin with a complete schedule already on hand, and a good initial solution allows several partial schedules to be discarded in the course of the tree search, simply because their bounds exceed the value of the initial solution.

3. COMPUTATIONAL EXPERIENCE

The algorithm was implemented as a FORTRAN program on the University of Michigan Terminal System (MTS), which uses an IBM 360/67 computer. The algorithm was tested on 90 different problems, which were specially designed to explore algorithm performance by varying problem size and varying tightness of the due-dates.

Three problem sizes were examined: $n=10$, 20, and 30. In each test problem, integer samples from a uniform distribution were used to construct the input data. The ready times were samples from a uniform distribution between 0 and 1,000. The processing times were independent of the ready times and were samples from a uniform distribution between 1 and $2{,}000/n$. Finally, the due-date for job i was taken from a uniform distribution between r_i and $r_i + 1{,}000K$, where K is a coefficient used for adjusting

the tightness of the due-dates. Three K-values were examined: $K=1.00$, 0.75, and 0.50. With three n-values and three K-values, there were nine different parametric configurations, and 10 test problems were generated for each configuration.

Because of limitations on the budget available for testing, we arbitrarily terminated the algorithm if it had not obtained an optimum solution after placing 1,000 nodes on the list. (This is a rather small number for a branch and bound solution: it is roughly half the total number of nodes in the whole branching tree for $n=6$.) Nevertheless, only 6 of the 90 problems failed to obtain an optimal solution. The average CPU times for the remaining 84 problems are displayed in Table 1. Broadly speaking, the tabulated data indicate that the algorithm is highly efficient in these problems. Permutation problems with 30 elements are usually prohibitively large for finding optimal solutions [2], but in 28 of our 30 test problems with $n=30$ an optimum was obtained in under 2.3 seconds. It is also interesting to note from Table 1 that run times are not very sensitive to K-values, even around $K=0.5$ where the due-dates are fairly tight.

TABLE 1. *Average CPU times in seconds. Superscripts indicate the number of problems requiring more than 1000 nodes in the branch and bound tree.*

K \ n	10	20	30	Overall Average
1.00	0.239	0.663	1.669	0.857
0.75	0.233	0.611[1]	1.758[1]	0.867
0.50	0.226[2]	0.722[1]	1.461[1]	0.803
Overall Average	0.233	0.665	1.629	
Maximum	0.303	0.926	2.291	

The algorithm exhibits some interesting behavior with regard to those problems in which more than 1,000 nodes are generated. For example, the two 10-job problems which generated 1,000 nodes were terminated at 4.1 and 6.0 seconds. All of the other 28 problems of the same size were solved in less than 0.31 seconds. Therefore there appeared to be a high probability that the solution time is very small and a small probability that the solution time is relatively quite large, but no observations occurred in between. The same behavior was observed for $n=20$ and 30. This behavior suggests that if computer time is at a premium, it might be wise to terminate the tree search if a solution has not been found in a moderate amount of time and then employ a heuristic procedure to find a schedule.

The speed of the algorithm directly reflects the number of nodes which need to be examined during the tree search. Table 2 shows the average number of nodes placed on the node list in each of the nine configurations for the problems in which optima were found. Again, we observe: (1) this statistic is not too sensitive to K, (2) there is a high probability that very few nodes will be examined, and (3) there were no instances in which the number of nodes generated lay between 132 and 1,000. The exceedingly small number of nodes usually observed reflects the value of the initial upper bound in limiting the size of the node list.

Table 3 displays the average CPU times per node and indicates an inverse relation with K and no perceptible effect of n.

TABLE 2. *Average Number of Nodes*

K \ n	10	20	30	Overall Average
1.00	14.8	40.4	84.9	46.7
0.75	11.1	35.4	91.9	46.1
0.50	8.9	39.5	72.8	40.4
Overall Average	11.6	38.4	83.2	
Maximum	22	78	132	

TABLE 3. *Average CPU Time Per Node*

K \ n	10	20	30	Overall Average
1.00	0.0161	0.0164	0.0197	0.0174
0.75	0.0209	0.0173	0.0191	0.0191
0.50	0.0254	0.0183	0.0201	0.0213
Overall Average	0.0208	0.0173	0.0196	

As mentioned above, the value of the heuristic initial solutions lies primarily in their role of allowing partial permutations to be discarded without ever being placed on the node list. The initial phase of the algorithm included four heuristic methods, which can be evaluated on their own merits as solution procedures. The heuristics were:

1. Ready time: sequence the jobs in nondecreasing order of their ready times, r_i.
2. Due-date: sequence the jobs in nondecreasing order of their due-dates, d_i.
3. Midpoint: sequence the jobs in nondecreasing order of the midpoints of their ready times and due-dates, $(r_i+d_i)/2$. Hence, use nondecreasing order of r_i+d_i.
4. PIO: sequence the jobs in the order of their initial appearance in the optimal preemptive schedule, which is constructed by the dynamic version of Jackson's Theorem.

The optimal values of $T_{\max}$ ranged from 0 to 703 in the 84 test problems which were solved. The absolute deviation of the heuristic solution from the optimal solution is one measure of the efficiency of a heuristic procedure, and these measures are displayed in Table 4. In addition, the frequencies

TABLE 4. *Performance of Heuristic Methods*

Heuristic	Average Deviation	Frequency Best [a]	Frequency Optimal [a]
Ready time	111.1	8	6
Due-date	73.4	31	25
Midpoint	47.7	36	27
PIO	34.6	50	24

[a] including ties

with which the individual procedures produced the best heuristic solution and the frequencies with which they produced an optimal solution are shown in Table 4. Of the four heuristics, PIO certainly appears to be the best, but the Midpoint heuristic is also quite good and is somewhat simpler to construct. It is interesting to note that the r_i-ordering and the d_i-ordering are substantially worse in terms of deviation from optimum than the Midpoint ordering, which uses both ready time and due-date information in forming a job sequence.

4. CONCLUSIONS

We have developed a branch and bound algorithm for minimizing maximum tardiness in the single-machine sequencing problem with early start times. The bound calculation for the nonpreemptive problem is based on the solution to the preemptive version of the problem and appears to be an efficient bounding procedure. In addition, further improvements in the algorithm have been obtained through the use of a substitution mechanism, an elimination mechanism, and an initial phase which produces a feasible solution with a simple set of heuristics.

The computational results indicate that problems with up to 30 jobs can be solved quite rapidly a high percentage of the time, although excessive computation times will occasionally arise. In addition, two very effective and simple heuristics were identified, and these may be useful in those situations where the branch and bound algorithm does not converge to an optimum rapidly enough. The significant feature of the algorithm, however, is that it can determine an optimum sequence so rapidly in so many cases. (Keep in mind that the potential number of sequences which may need to be examined in a 30-job problem is over 2×10^{32}.)

A closely related problem has been treated by Bratley, Florian, and Robillard [1]. In their work, the objective is to determine whether a schedule exists in which $T_{\max} = 0$, and if so to minimize the total elapsed time of the schedule. They also developed a branch and bound approach and observed computational behavior quite like that of our algorithm. Another treatment of the $T_{\max}$ problem has been described by Dessouky and Margenthaler [3], but they have not yet reported the computational behavior of their approach.

REFERENCES

[1] Bratley, P., M. Florian, and P. Robillard, "Scheduling with Earliest Start and Due Date Constraints," Nav. Res. Log. Quart. *18*, 511–519 (Dec. 1971).

[2] Conway, R., W. Maxwell, and L. Miller, *Theory of Scheduling* (Addison–Wesley, Reading, Mass., 1967).

[3] Dessouky, M. I. and C. R. Margenthaler, "The One-Machine Sequencing Problem with Early Starts and Due Dates," *AIIE Transactions*, Vol. 4, No. 3 (Sept. 1972).

[4] Jackson, J. R., "Scheduling a Production Line to Minimize Maximum Tardiness," Research Report #43, Management Science Research Project, UCLA (Jan. 1955). See also [2] above.

SOME SIMPLE SCHEDULING ALGORITHMS

W. A. Horn

National Bureau of Standards

ABSTRACT

This paper considers situations in which jobs require only one operation on a single machine, or on one of a set of identical machines. Penalty-free interruption is allowed. Some simple algorithms are given for finding optimum schedules to minimize maximum lateness and total delay, for the single-machine case, and maximum lateness for a restricted multimachine case. A simple flow problem formulation permits minimizing maximum lateness for the more general multimachine case.

1. INTRODUCTION

We present some algorithms to minimize maximum lateness and total delay for single-machine and multimachine problems concerned with the scheduling of single-operation jobs. Despite their simplicity these specific algorithms have apparently not been published previously.

The situation is as follows: Each job i becomes available for processing at some time $a_i \geq 0$ and requires d_i time units for completion. In the multimachine case, all machines are assumed to have identical characteristics. Job splitting without penalty is allowed, so that a feasible solution must include one or more intervals of service to job i, each interval on a single machine, whose total length is d_i. In the case where maximum lateness is to be minimized an additional quantity b_i, the due date of job i, is also specified.

The next two sections deal with the single machine case, while the last section treats the multimachine case.

2. MINIMIZING MAXIMUM LATENESS: ONE MACHINE

It was shown by Jackson in 1955 [3] that if all jobs are available at time 0 and each job i has a due date b_i, then the maximum lateness, for the single machine case, is minimized by performing the jobs in the order of increasing b_i. (See also [2], pp. 30 f.) The following algorithm provides a similar minimization for generally different a_i. This solution method is somewhat more general, and perhaps simpler, than a method presented in [1].

ALGORITHM: Let E be the subset of all jobs with earliest time of availability, which we designate A_1. Let A_2 be the next earliest time of availability (or $A_2 = \infty$ if all jobs are available at A_1). Let job i be a job in E which has earliest due date b_i of all jobs in E. Let $L_1 = \min \{d_i, A_2 - A_1\}$. Assign job i to the interval $[A_1, A_1 + L_1]$. If job i is not completed, reduce d_i by L_i, otherwise drop job i from the list. Repeat this operation for all remaining jobs, stipulating a minimum availability time of $A_1 + L_1$ for all remaining jobs. Continue until all jobs are completely assigned.

The proof that this algorithm minimizes maximum lateness is as follows. Let $I \rightarrow J$ denote the fact that interval J directly follows interval I. That is, $\max I = \min J$. Suppose that S is an optimum schedule for the jobs, and let the time interval $[A_1, A_1 + L_1]$, defined above, be composed in S of the intervals $I_1, I_2, \ldots, I_m$, where $I_1 \rightarrow \ldots \rightarrow I_m$, and each I_j represents an interval of service

to one job or an interval of nonservice. If I_1 is not for servicing job i, the job chosen by the algorithm, but rather job k, then $k \epsilon E$. Change the schedule so that job i is serviced during I_1 (where $|I_1| \leq L_1 \leq d_i$) by servicing k, rather than i, during any other intervals in S of total length $|I_1|$ in which i was originally serviced. Since the due date of job i is at least as early as that of job k, it is clear that such a change does not increase the overall maximum lateness of the schedule. (If I_1 is idle time, simply move $|I_1|$ units of service for job i to I_1.)

Repeat the above process for the subintervals $I_2, I_3, \ldots, I_n$, replacing any servicing of a job other than i, or any idle interval, by a servicing of i. By the same reasoning, overall maximum lateness is not increased. Thus a new optimal schedule S_1 is created in which i is serviced during all of $[A_1, A_1 + L_1]$.

It is clear that by dropping $[A_1, A_1 + L_1]$ from consideration now and reducing d_i to $d_i - L_1$ we may repeat the above process on $[A_1 + L_1, \infty)$ in a way corresponding to a repeated application of the algorithm, without increasing maximum lateness, so that finally an optimal schedule is obtained which is that given by the algorithm.

3. MINIMIZING TOTAL DELAY: ONE MACHINE

In this problem no due dates are given. The object is to minimize $\sum_i (f_i - a_i)$, where f_i is the time at which job i is finished. If all jobs are available to be serviced at the same time, the problem is very simply solved, merely by taking the jobs in the order of increasing d_i. (Cf. [2], pp. 26 ff., for example.) With general a_i, and job splitting allowed, a similar method applies.

ALGORITHM: Let E be the set of all jobs with earliest time of availability, denoted by A_1. Let A_2 be the next earliest time of availability (where $A_2 = \infty$ if all jobs are available at A_1). Let i be a job in E with smallest d_i of all jobs in E, and let $L_1 = \min \{d_i, A_2 - A_1\}$. Assign job i to the interval $[A_1, A_1 + L_1]$. If job i is not finished during this interval, reduce d_i by L_1; otherwise, drop job i from the list. Repeat this operation for all remaining jobs, stipulating that no job shall have an availability time earlier than $A_1 + L_1$. Continue to apply it until all jobs are serviced.

The proof that the algorithm minimizes delay is as follows. Suppose that S is an optimum schedule, and let the interval $[A_1, A_1 + L_1]$ be composed of subintervals $I_1, I_2, \ldots, I_m$ of service to individual jobs or nonservice, where $I_1 \rightarrow I_2 \rightarrow \ldots \rightarrow I_m$. If I_1 is not for servicing i, the job chosen by the algorithm, but rather job k, then $k \epsilon E$. Consider all those intervals in S, denoted $I_1 = J_1 < J_2 < \ldots < J_r$, (where $J_{j-1} < J_j$ means simply that J_{j-1} precedes J_j), such that either i or k is serviced during each J_j. Rearrange the schedule S so that the first d_i time units of UJ_j (determined by taking an initial subsequence $J_1, J_2, \ldots$, and perhaps a partial last interval, so that total length is d_i) are devoted to servicing i, and the last d_k time units (since $\sum_j |J_j| = d_i + d_k$) are devoted to job k.

In this new schedule, S', the later finishing of jobs i and k, namely job k, is finished at the end of J_r, say at time t_r as was the finish time for i or k in the original schedule S. The earlier-finishing job, i, is finished at some time t' in the new schedule, no later than the finish time of the earlier-finishing job in S, say time t, since $d_i \leq d_k$ and the servicing of i was always done at the earliest possible time in S'. Therefore, the total delay for i and k in S', namely $(t_r - a_k) + (t' - a_i)$, is at least as small as the total delay in S, namely $(t_r - a_k) + (t - a_i)$ or $(t_r - a_i) + (t - a_k)$, depending on which job finished first in S. Thus S' does not increase total delay time.

As in the previous case, repeated changes in the schedule assign all of $[A_1, A_1 + L_1]$ to servicing

i, without increasing total delay, and similar repetitions show that the order of servicing given by the algorithm does not increase delay and hence is optimal.

4. MULTIMACHINE PROBLEMS

For the multimachine case, little seems to be known with regard to simple minimization algorithms. When all jobs are available simultaneously, the machines are identical, and there is no constraint on the time to finish, then a method is known [2] to ensure minimum overall delay, as follows. Order the jobs for increasing d_i-values as $i(1), i(2), \ldots, i(n)$. If there are m machines, assign each of the first m jobs in the sequence to first place on each of the machines, each of the next m jobs to second place, and so on. Unfortunately this method does not seem to generalize to cases with varying a_i-values, nor do the algorithms of sections 2 and 3 seem to generalize to multimachine situations. Therefore it is necessary to attack these problems on an individual basis.

The algorithms presented here apply to minimizing maximum lateness, but they will be presented in terms of satisfying feasibility constraints, that is, finishing each job i by time b_i. Then it will be shown how these two formulations are equivalent.

Consider first the case where all jobs have $a_i = 0$, $b_i = T > 0$, and there are m machines. For this case we have:

THEOREM 1: If there are m machines and n jobs, with all $a_i = 0$, $b_i = T > 0$, and if job splitting is allowed, but no job may be processed on two machines simultaneously, then there exists a schedule which finishes all jobs by T if, and only if

(a) $d_i \leq T$ for each job i; and

(b) $\sum_{i=1}^{n} d_i \leq mT$.

PROOF: The proof of necessity is obvious. To prove sufficiency, construct a circle of circumference T, and identify some point p on the circumference as the "initial" point. Starting at p and going clockwise, lay off intervals of lengths, $d_1, d_2, d_3, \ldots, d_n$. By condition (a), no such interval overlaps itself. By (b), the intervals circle the circumference no more than m times.

Now identify the circumference with m copies of the interval $[0, T]$ as follows. Let a point x, at distance d from p moving clockwise, be identified with $d \in [0, T]$. Identify each interval of length d_i with the one or two intervals in $[0, T]$ corresponding to it. Whenever an interval is broken (passes point p), put the earlier half of it on the next $[0, T]$ interval. Finally, associate each $[0, T]$ interval with a machine. Since no d_i interval on the circle overlaps itself, there is no simultaneous processing of the same job on two machines, and so the assignment is feasible. This proves sufficiency.

Theorem 1 allows us to prove more easily a theorem about the case where not all jobs must be completed by the same time. The proof of theorem 2, as for theorem 1, will be constructive in nature, thereby providing the natural basis for an algorithm.

Suppose all $a_i = A$, and let $\{e_i\}_0^k$ represent the set of b_i's without repetition, together with A, where

$$A = e_0 < e_1 < \ldots < e_{k-1} < e_k \qquad (k \leq n). \tag{4.1}$$

Let $I_j = [e_{j-1}, e_j]$ and $J_j = [e_0, e_j]$. For each J_j we define a quantity $r(i, j)$ which in a sense describes

the amount of work which *must* be performed on job i during J_j if feasibility is to be obtained. Specifically, let

$$\begin{aligned} r(i,j) &= \min\{\max\{0,\ d_i-(b_i-e_j)\},\ d_i\} \\ &= \min\{\max\{0,\ d_i-b_i+e_j\},\ d_i\}. \end{aligned} \tag{4.2}$$

Then clearly

$$0 \leq r(i,j) \leq \begin{cases} d_i \\ e_j - e_0, \end{cases} \tag{4.3}$$

where $r(i,j) \leq e_j - e_0$ because $d_i \leq b_i - e_0$ in any problem which is to be feasible.

If $e_j > b_i$, then $r(i,j) = d_i$, indicating, of course, that all work on job i must be done in J_j. If $e_j \leq b_i$, but $d_i > b_i - e_j$ (i.e., the amount of work on job i is greater than the time remaining after J_j in which to do it), then $r(i,j) = d_i - b_i - e_j$. If $d_i \leq b_i - e_j$, then all of the work on job i may be performed outside J_j, so that $r(i,j) = 0$.

Define

$$N_j = \sum_{i=1}^{n} r(i,j) \qquad (1 \leq j \leq k). \tag{4.4}$$

It is clear that, if feasibility is to be obtained, the work required to be done in J_j, namely N_j, must not exceed the capacity of the machines and so

$$N_j \leq m(e_j - e_0). \tag{4.5}$$

The discussion leads us to the following result.

THEOREM 2: Let there be m machines and n jobs, with all $a_i = A$. If job splitting is allowed but no job may be processed on two or more machines simultaneously, then there exists a schedule which finishes all jobs on time if and only if

(a) for each i, $d_i \leq b_i - A = b_i - e_0$; and
(b) for each j, (4.5) holds.

PROOF: Necessity has already been shown. The proof of sufficiency is inductive in nature (as well as constructive).

For $k = 1$, the one-interval case, each $r(i, 1) = d_i$, $N_1 = \Sigma_{i=1}^{n}\ d_i$, and so (4.5) becomes

$$\sum_{i=1}^{n} d_i \leq m(e_1 - e_0). \tag{4.6}$$

Clearly this is a case of Theorem 1 (with an adjustment of parameters), and so conditions (a) and (b) are sufficient for all m and n.

Now let $k>1$, and assume the proof has been given for all cases less than k. Consider the quantities

$$\bar{r}(i,j)=\min\{r(i,j),\ e_1-e_0\} \qquad (\text{all } i,j), \tag{4.7}$$

and the sums

$$\bar{R}_j=\sum_{i=1}^{n}\bar{r}(i,j) \qquad (\text{all } j). \tag{4.8}$$

Let

$$\bar{j}=\max\{j:\bar{R}_j\leq m(e_1-e_0)\}. \tag{4.9}$$

Since, for each i,

$$r(i,1)\leq r(i,2)\leq\ \ldots\ \leq r(i,k), \tag{4.10}$$

we have, from (4.7),

$$\bar{r}(i,1)\leq\bar{r}(i,2)\leq\ \ldots\ \leq\bar{r}(i,k) \tag{4.11}$$

and so

$$\bar{R}_{\bar{j}}\leq\bar{R}_j\leq m(e_1-e_0) \qquad (j\leq\bar{j}). \tag{4.12}$$

Define quantities w_i as follows. If $\bar{j}=k$, then $w_i=\bar{r}(i,k)$. If $\bar{j}<k$, then the w_i's are any numbers such that

$$\bar{r}(i,\bar{j})\leq w_i\leq\bar{r}(i,\bar{j}+1) \tag{4.13}$$

and

$$\sum_{i=1}^{n}w_i=m(e_1-e_0). \tag{4.14}$$

(Note that $\bar{j}\geq 1$, since otherwise condition (b) would be violated for $j=1$.)

Next assign w_i as the amount of work to be done on job i in $[e_0, e_1]$, where the scheduling of such work is accomplished by the application of Theorem 1. (Relation (4.7), and (4.13) if $\bar{j}<k$, imply that condition (a) of Theorem 1 is satisfied. That condition (b) is satisfied follows from (4.14) if $\bar{j}<k$, and from (4.8–4.9) if $\bar{j}=k$.) Reduce the quantities d_i to

$$\bar{d}_i=d_i-w_i \tag{4.15}$$

for the remaining problem which has one fewer interval. Then $\bar{d}_i$ represents the amount of work to be done on job i during the remaining time interval, $[e_1, e_k]$. (Note that $\bar{d}_i\geq 0$, since if $\bar{j}<k$ then $d_i\geq$

$\bar{r}(i, \bar{\jmath}+1) \geq w_i$, while if $j=k$ then $d_i \geq \bar{r}(i, k)=w_i$.) Then it is asserted that the remaining problem satisfies constraints (a) and (b), where e_1 is substituted for e_0 and $\bar{d}_i$ for d_i. If this is the case, then the theorem is proved, since the above scheduling in the interval $[e_0, e_1]$ can be joined to the scheduling in $[e_1, e_k]$ whose existence is implied by the inductive assumption, thereby giving a feasible scheduling for the entire problem.

Consider first condition (a). If $d_i \leq b_i - e_1$, then $\bar{d}_i \leq d_i \leq b_i - e_1$, satisfying (a). If $d_i > b_i - e_1$, then, since $b_i \geq e_1$, $r(i, 1) = d_i - (b_i - e_1)$ by (4.2), and so

$$\begin{aligned} \bar{d}_i &= d_i - w_i \leq d_i - \bar{r}(i, \bar{\jmath}) \leq d_i - \bar{r}(i, 1) \\ &= d_i - r(i, 1) \qquad \text{by (4.3) and (4.7)} \\ &= b_i - e_1, \end{aligned} \tag{4.16}$$

so that condition (a) holds in this case also.

Next consider condition (b). For all i and j, let

$$r'(i, j) = \min \{\max \{0, \bar{d}_i - b_i + e_j\}, \bar{d}_i\}, \tag{4.17}$$

the new $r(i, j)$ −values, and let

$$N'_j = \sum_{i=1}^{n} r'(i, j). \tag{4.18}$$

We now show that

$$r'(i, j) = \max \{0, r(i, j) - w_i\} \tag{4.19}$$

for each i and j.

Suppose first that $e_j \geq b_i$. Then $r(i, j) = d_i$, from (4.2), while $r'(i, j) = \bar{d}_i = d_i - w_i$, from (4.17), and so (4.19) holds. Next let $e_j < b_i$.

Then

$$r(i, j) = \max \{0, d_i - b_i + e_j\} \tag{4.20}$$

while

$$r'(i, j) = \max \{0, d_i - w_i - b_i + e_j\}. \tag{4.21}$$

If $d_i - b_i + e_j \geq w_i$, then clearly $r'(i, j) = r(i, j) - w_i$.
If $d_i - b_i + e_j < w_i$, then $r'(i, j) = 0 > r(i, j) - w_i$.
Thus (4.19) again holds.

Returning to the proof of (b), we consider the cases $j \leq \bar{\jmath}$ and $j > \bar{\jmath}$. If $j \leq \bar{\jmath}$, let

$$S_j = \{i : r'(i, j) > 0\}. \tag{4.22}$$

By (4.19), for every $i \in S_j$

$$r(i, j) = r'(i, j) + w_i \geq r'(i, j) > 0, \tag{4.23}$$

so that

$$w_i \geq \bar{r}(i, \bar{j}) \geq \bar{r}(i, j) > 0 \tag{4.24}$$

from (4.13) and (4.11), while

$$w_i < r(i, j) \tag{4.25}$$

from (4.23). By (4.7), $\bar{r}(i, j) = e_1 - e_0$ (since otherwise $\bar{r}(i, j) = r(i, j)$, contradicting (4.24) and (4.25), and so

$$w_i \geq \bar{r}(i, j) = e_1 - e_0. \tag{4.26}$$

But if $\bar{j} < k$ then $w_i \leq \bar{r}(i, j+1) \leq e_1 - e_0$, by (4.13) and (4.7), while if $\bar{j} = k$ then $w_i = \bar{r}(i, k) \leq e_1 - e_0$ by (4.7), so in either case

$$w_i = e_1 - e_0. \tag{4.27}$$

Therefore, by the constraint (4.5) on $[e_0, e_1]$, there are no more than m members in S_j. Thus

$$\begin{aligned} \sum_{i=1}^{n} r'(i, j) &= \sum \{r'(i, j) : i \epsilon S_j\} \\ &= \sum \{r(i, j) - w_i : i \epsilon S_j\} \\ &\leq m(\max_i r(i, j) - (e_1 - e_0)). \end{aligned} \tag{4.28}$$

But from (4.3),

$$\max_i r(i, j) \leq e_j - e_0, \tag{4.29}$$

so that

$$\sum_{i=1}^{n} r'(i, j) \leq m(e_j - e_1), \tag{4.30}$$

and so inequality (4.5) is satisfied for $j \leq \bar{j}$ in the new problem.

If $j > \bar{j}$, then $\bar{j} < k$ so that (4.14) holds. From (4.13), we have

$$r(i, j) \geq \bar{r}(i, j) \geq \bar{r}(i, \bar{j}+1) \geq w_i, \tag{4.31}$$

for each i, so that

$$r'(i, j) = r(i, j) - w_i, \tag{4.32}$$

from (4.19). Thus

$$\sum_{i=1}^{n} r'(i,j) = \sum_{i=1}^{n} r(i,j) - \sum_{i=1}^{n} w_i$$

$$= \sum_{i=1}^{n} r(i,j) - m(e_1 - e_0) \tag{4.33}$$

$$\leq m(e_j - e_0) - m(e_1 - e_0)$$

$$= m(e_j - e_1).$$

Thus (4.5) is satisfied also for $j > \bar{\jmath}$, and the theorem is proved.

Theorem 2 can easily be used to find a schedule which minimizes maximum lateness, in the following manner. If every b_i is increased (or decreased) by adding some fixed quantity M, then a feasible solution for the due dates $\{b_i + M\}$ is equivalent to a solution for the original problem which has maximum lateness no greater than M. Clearly it is desired to find the minimum M for which a feasible solution exists.

But this problem is the same as simply changing A, the availability time, to $A - M$, with the original set $\{b_i\}$. It is easy to find a minimum M such that conditions (a) and (b) hold. For (a) is simply solved, and changing A to $A - M$ merely adds mM to the right side of each inequality (4.5). In fact the minimum value of M to satisfy (b) is given as

$$M = \max_j \{N_j/m - e_j\} + e_0, \tag{4.34}$$

while the minimum value to satisfy (a) is

$$M = \max_i \{d_i - b_i\} + A. \tag{4.35}$$

The larger of the two values is the one used.

Theorem 2 can also be used to solve the scheduling problem where all $b_i = T$, some fixed value, while the a_i are variable. To see this, we note that there is a scheduling which fits all of the work for job i into the time interval $[a_i, T]$ if and only if there is a schedule which fits jobs of equal duration (same d_i) into the intervals $[-T, -a_i]$. This latter problem is then handled by Theorem 2.

5. A LINEAR PROGRAM FOR THE MULTIMACHINE CASE

In this section we consider the more general multimachine problem where maximum lateness is to be minimized and all the conditions of Theorem 2, section 4, are satisfied, except that *both* the a_i and the b_i may take on different values.

For a given problem, let the a_i and b_i be ordered as

$$e_0 < e_1 < \ldots < e_k.$$

Using a technique similar to that of [1], define a flow network with k sources S_j, representing the k intervals $[e_{j-1}, e_j]$, where each S_j has an outflow limited to $m(e_j - e_{j-1})$. There are also n sinks T_i, representing the n jobs. If any job i can feasibly be serviced in $[e_{j-1}, e_j]$, i.e., $b_i > e_{j-1}$ and $a_i < e_j$, then

draw an arc A_{ij} from S_j to T_i, where A_{ij} will have upper capacity $(e_j - e_{j-1})$. Furthermore, sink T_i will have a required inflow of d_i units.

The problem of finding a feasible flow in the above network will, when solved, give a feasible solution to the multimachine problem. For Theorem 1 of section 4 may be used to find an actual scheduling in $[e_{j-1}, e_j]$, given the amounts of time in this interval assigned to the various jobs. These latter quantities are the flows in the arcs A_{ij}.

The above procedure, which determines whether a feasible schedule exists, may be used to determine maximum lateness in an optimal schedule, by a process similar to that of section 4. The value of each b_i is increased (or decreased) by some fixed amount, and the flow problem is solved to determine whether a feasible solution exists. (This will involve a new set of e_j each time, of course.) By trial and error, the minimum value to be added to each b_i is found.

REFERENCES

[1] Bratley, P., M. Florian, and P. Robillard, "Scheduling with Earliest Start and Due Date Constraints," Nav. Res. Log. Quart. *18*, 511–519 (1971).

[2] Conway, R., W. L. Maxwell, and L. W. Miller, *Theory of Scheduling* (Addison Wesley, 1967).

[3] Jackson, J. R., "Scheduling a Production Line to Minimize Maximum Tardiness," Mgt. Sci. Res. Project, Resch. Rept. 43, UCLA (Jan. 1955).

AN EXPERIMENTAL COMPARISON OF SOLUTION ALGORITHMS FOR THE SINGLE-MACHINE TARDINESS PROBLEM

Kenneth R. Baker

Department of Industrial Engineering
North Carolina State University

and

James B. Martin

Department of Industrial and Operations Engineering
University of Michigan

ABSTRACT

A basic problem in scheduling involves the sequencing of a set of independent tasks at a single facility with the objective of minimizing mean tardiness. Although the problem is relatively simple, the determination of an optimal sequence remains a challenging combinatorial problem. A number of algorithms have been developed for finding solutions, and this paper reports a comparative evaluation of these procedures. Computer programs for five separate algorithms were written and all were run on a data base designed to highlight computational differences. Optimizing algorithms developed by Emmons and by Srinivasan appeared to be particularly efficient in the comparative study.

1. INTRODUCTION

A basic problem in scheduling involves the sequencing of a set of jobs at a single machine with the objective of minimizing mean job tardiness. Although the statement of the problem is relatively simple, it is often a difficult matter to find good solutions quickly. A number of tardiness-oriented algorithms for the single machine problem have been developed [2, 3, 4, 6, 9, 10], and this paper reports a comparative evaluation of these procedures.

The sequencing model consists of a set of n jobs which are simultaneously available (at time zero) for processing by a single machine. Job j has an associated processing time (p_j) and a due-date (d_j), both of which are known in advance. The machine is continuously available, and job setup times are independent of sequence and are included in the processing times. In a given sequence, job j will have a completion time (C_j) and an associated tardiness (T_j) defined by

$$T_j = \max \{0, C_j - d_j\}.$$

The objective is then minimization of mean job tardiness ($\bar{T}$), where

$$\bar{T} = \frac{1}{n} \sum_{j=1}^{n} T_j.$$

(An equivalent objective is to minimize the total tardiness in the sequence.) Under these conditions, there exists an optimal schedule in which no job is preempted [1]. It is therefore sufficient to examine

at most $n!$ different sequences—corresponding to all permutations of the n jobs in sequence—in the search for an optimal solution.

The following section briefly discusses each of the algorithms included in the comparative study. The reader should refer to the original sources for full details.

2. THE ALGORITHMS

2.1 Dynamic Programming

The earliest available algorithm for the tardiness problem was dynamic programming [4, 6]. This controlled enumeration approach is very general and applicable to more complicated scheduling problems than the single machine tardiness problem. Until recently, it was the only reported method short of complete enumeration which could guarantee optimality.

As in [6], let I denote the set of all n jobs; let J denote some subset of the jobs; and let J' denote the complement of J. Also, let $q(J')$ denote the total processing time required by the jobs in J'. Now consider a sequence in which all the jobs in J' are processed before any job in J is begun. Let $f(J)$ denote the minimum total tardiness associated with a sequence of the jobs in J. The basic recursion relation is:

$$f(J) = \min_{j \in J} \{\max [0, q(J') + p_j - d_j] + f(J - \{j\})\}. \tag{1}$$

The optimal value of total tardiness is $f(I)$. To calculate $f(I)$, the procedure begins with $f(\phi) = 0$ and computes $f(\cdot)$ for subsets of size 1. With these results it can then calculate $f(\cdot)$ for subsets of size 2, and so on, calculating $f(\cdot)$ on larger and larger subsets until $f(I)$ is obtained. With this approach, the dynamic programming procedure requires that the function $f(\cdot)$ be determined for each of 2^n different sets. As many authors have observed, the computation times and storage requirements of the algorithm grow exponentially with problem size and render the algorithm impractical for large problems.

Perhaps one virtue of the dynamic programming procedure is that for a given problem size, the time to find an optimal solution on a computer is essentially deterministic, since the number of calculations in the procedure is fixed. Because of this predictability, and because the method has wider usefulness in solving combinatorial problems, dynamic programming provides a meaningful reference point for comparison.

2.2 Branch and Bound

The computational effort required by the dynamic programming algorithm can generally be reduced using a branch and bound approach, such as the one described by Elmaghraby [2] for a more general problem. For each subset J in the dynamic programming formulation,* it is possible to calculate an associated lower bound $B(J)$ on the total tardiness which would be obtained for a schedule in which the jobs in set J are preceded by all the jobs in set J'. (The value of $f(J)$ itself is such a bound.) At any stage in the calculations if $J_1 \subset J_2$ and $f(J_1) \geq f(J_2)$, then by Elmaghraby's Theorem 1 it is not necessary to consider any further completions of the partial sequence represented by subset J_1. In particular, when J_2 contains all n jobs a feasible solution exists which is at least as good as any completion of the partial sequence represented by J_1. The branching procedure terminates when no uncompleted

*This statement implies that a subset J containing k elements may actually be considered k different times, according to which of its elements comes first. Thus the recursion relation (1) requires that set J be evaluated k "different" ways.

subset has a bound less than the value of an on-hand feasible sequence, for at this stage the on-hand sequence is optimal.

The method's efficiency depends on the tightness of the lower bounds. Elmaghraby suggested one simple way to improve on $f(J)$ as a lower bound. Let $k \epsilon J'$ denote the job with the latest due-date in set J'. Then a lower bound is

$$B(J) = f(J) + \max [0, q(J') - d_k]. \tag{2}$$

The branch and bound procedure maintains a list of active subsets ranked by their bounds. At each stage, the subset J at the top of the list (the subset with the lowest bound) is removed and replaced with several subsets, each of which is formed by adding one new element to J. As stated in Elmaghraby's Lemma 1, if one job can be added to J without incurring more tardiness, then it is sufficient to replace set J with just one augmented set. Therefore Lemma 1 and Theorem 1 provide mechanisms for curtailing the dynamic programming calculations.

Branch and bound methods have become important tools in the solution of large combinatorial problems and especially scheduling problems (see, for example, chapters 4, 5, and 6 of Reference 1). The main purpose of the branch and bound approach is to control effectively the enumeration of partial solutions en route to determining an optimum. In this sense it is a similar approach to dynamic programming, except that the behavior of branch and bound programs is much less predictable. Both storage requirements and computation times may vary drastically for problems of a given size. This variability is a function of the numbers in a specific problem and of the efficiency of the lower bounds.

2.3. The Emmons Algorithm

Emmons [3] derived special conditions under which it is possible to directly sequence some of the jobs and reduce the size of the problem. Let

A_j = the set of jobs which are known to come after job j in an optimal sequence.

B_j = the set of jobs which are known to come before job j in an optimal sequence.

A_j' = the complement of set A_j.

$j \rho k$ denote the following relation between jobs j and k.

$j \rho k$ if and only if (1) $p_j < p_k$, or

(2) $p_j = p_k$ and $d_j \leq d_k$.

Emmons proved that . . .

THEOREM 1: If $j \rho k$ and $d_j \leq \max \left\{ d_k, p_k + \sum_{i \epsilon B_k} p_i \right\}$, then $j \epsilon B_k$.

THEOREM 2: If $j \rho k$ and $d_j > \max \left\{ d_k, p_k + \sum_{i \epsilon B_k} p_i \right\}$

and $d_j + p_j \geq \sum_{i \epsilon A_k'} p_i$, then $j \epsilon A_k$.

THEOREM 3: If $j \rho k$ and $d_k \geq \sum_{i \epsilon A_j'} p_i$, then $j \epsilon A_k$.

When Theorem 1 is used, for example, the algorithm begins with $B_k = \phi$ and attempts to add jobs to it. With each added job, the conditions change and more jobs might be added to B_k. The algorithm pro-

ceeds recursively and attempts to place $(n-1)$ jobs in the set B_k. If this can be accomplished, job k can be last in an optimal sequence and can be removed from the original problem.

The Emmons algorithm first attempts to assign a job to the last position in sequence. If this can be done, it attempts to assign the next to last position, and so on until no job can be removed from the problem and placed last. Next, the algorithm attempts to assign the first position and continues to construct the beginning of the sequence until no job can be removed from the problem and placed first. These two routines may construct a complete sequence, in which case an optimal solution has been found. If some positions in the middle of the sequence remain unfilled, Emmons' algorithm follows a branching procedure. Each branch corresponds to imposing a precedence restriction on a pair of unsequenced jobs (i.e., job i must come before job j, so that $i \epsilon B_j$ and $j \epsilon A_i$, where jobs i and j are yet unsequenced.) An alternate branch imposes the opposite precedence restriction. Having partitioned the problem into smaller subproblems, the algorithm proceeds to a full solution, branching again whenever necessary.

Emmons did not test a computerized version of this algorithm, and the present study may be the first attempt to explore its computational properties. As with any branching routine, the computational effort required for a given problem size depends on the specific problem data. Although the desirability of the branching mechanism is open to question, there is no doubt that the basic advantage in the Emmons approach is its ability to reduce the size of the original problem. One important facet of the current study is an evaluation of the effectiveness of this reduction mechanism.

2.4 The Srinivasan Algorithm

Srinivasan [9] proposed a three-phase hybrid algorithm which incorporates the Emmons precedence relationships into a dynamic programming framework. Phase 1 places last in the optimal sequence all jobs j for which $d_j \geq \sum_{i=1}^{n} p_i$. Phase 2 uses Theorems 1 and 2 of Emmons to develop a partial ordering of the jobs and, if possible, to identify the first and last few jobs of the optimal sequence. Phase 3 optimally arranges the remaining jobs using a dynamic programming algorithm modified to make use of the partial ordering developed in Phase 2. The computations required under a normal dynamic programming formulation can be reduced by the use of the following four propositions:

PROPOSITION 1: At stage k, those jobs j for which $|A_i| \geq k$ need not be considered* in (1).

PROPOSITION 2: At stage k, for those J which contain a job j with $|A_j| = k-1$, job j must come first in the optimal order.

PROPOSITION 3: At stage k, for the subsets of J which contain a job j with $|A_j| < k-1$, only the J which contain all $j \epsilon A_j$ need be considered.

PROPOSITION 4: At stage k, those jobs j for which $|B_j| \geq n-k+1$ need not be considered for the first position in sequence.

Srinivasan did not report how solution times would be affected if only some of these propositions were implemented. It is possible that the time spent verifying the conditions of Propositions 1–4 more than offsets the time gained by curtailing the dynamic programming calculations.

2.5 The Wilkerson-Irwin Algorithm

Wilkerson and Irwin [10] described a two-phase heuristic algorithm. Phase 1 employs a decision rule to manipulate lists of scheduled and unscheduled jobs until a full sequence has been constructed. Phase 2 examines a small set of possible job interchanges in an effort to improve the on-hand solution.

*The notation $|A_i|$ signifies the number of jobs in set A_i.

The sequence generated by Phase 1 may or may not be optimal. Wilkerson and Irwin proved that a sufficient condition for optimality is that the Phase 1 sequence contains no overlapping tardiness intervals. (In other words, there exists no point in the sequence at which two jobs are already tardy and remain uncompleted.) Unfortunately, this is only a sufficient condition. Therefore, it is possible for Phase 1 to produce a sequence which does not satisfy the condition, but which is still optimal. In view of the speed of the Wilkerson-Irwin heuristic, the present study has investigated three aspects of general interest:

1. How often does Phase 1 produce an optimum sequence?
2. How often does Phase 1 produce an optimum sequence which can be recognized (via the sufficient condition) as optimal?
3. What is the trade-off between speed of solution and suboptimality?

3. THE TEST DATA

The test data were designed primarily to examine the relative performance of the algorithms. The approach involved solving a number of test problems by using each of the solution methods. In addition, the test problems were designed to examine special characteristics of problem structure and their effects on algorithm performance. These characteristics are discussed in the following sections.

3.1 Problem Size

The basic disadvantage of dynamic programming is that the solution times tend to grow exponentially with the size of the problem. The other approaches tend to be more efficient than dynamic programming, and two questions regarding their efficiencies are of general interest:

1. Which algorithm is most rapid, and does relative solution time depend significantly on problem size?
2. What inferences appear logical about the relative behavior of the algorithms for larger problems?

The test data included problem sets of size $n=8$, 10, 12, and 15. In the experiments, job sets of size 15 were sufficient to yield dramatic differences in solution times among most of the methods. It is also worth noting that with many computers, solutions by dynamic programming would be prohibitively costly for problems of size 15. The four different problem sizes studied provided an opportunity to discover aspects of algorithm behavior highly sensitive to problem size.

3.2 Tardiness Factor

The tardiness factor of a problem [9] is a coarse measure of the proportion of jobs which might be expected to be tardy in an arbitrary sequence. If $\bar{p}$ denotes the average processing time in the job set then, in some average sense, a number k of the jobs can be processed by time $k\bar{p}$. If $\bar{d}$ denotes the average due-date then k jobs will be completed on time, in some average sense, when $\bar{d}=k\bar{p}$. Hence if t is defined to be the tardiness factor, it satisfies:

$$\bar{d}=n(1-t)\bar{p}. \tag{3}$$

Srinivasan found that his hybrid algorithm was least efficient for t values around 0.6. The test problems in the present study were designed to explore two tardiness factor regions, $t=0.2$ and 0.6, where it might be anticipated that the hybrid algorithm would be more efficient for the lower value.

In the test problems, the processing times were samples drawn from a normal distribution with mean $\mu_p=100$ and standard deviation 25; and the due-dates were samples from a uniform distribution

with mean μ_d determined by the tardiness factor. For example, when the due-dates were independent of the processing times,

$$\mu_d = n(1-t)\mu_p. \tag{4}$$

When due dates were dependent on processing times, a processing time, p, was generated first, then the due-date was chosen from a distribution for which

$$\mu_d = n(1-t)p. \tag{5}$$

The purpose of including the tardiness factor in the construction of test problems was to determine whether the efficiencies of the different algorithms depend on the tardiness factor.

3.3 Due-date Range

Wilkerson and Irwin [10] observed that the behavior of their algorithm was related to the range of the due-dates in the job set. When this range was above 0.95 of the sum of processing times, their Phase 1 was highly effective, but much less likely to produce optima when the range was below 0.85 of the sum of processing times.

In the test problems where due-dates and processing times were independent, the due-dates were samples from a uniform distribution on the interval (a, b). The mean of this distribution was determined by the tardiness factor (Equation (4)) and the range, R, of this distribution satisfied

$$R = \frac{b-a}{n\mu_p}. \tag{6}$$

Hence, R represents the maximum possible due-date range in a given test problem, expressed as a fraction of the expected value of the length of the schedule. In the dependent case, the range constraint (Equation (6)) was applied only to the full set of due-dates so that the range of the uniform distribution that applied to individual samples was varied as a function of the value of p obtained. As in the case of the tardiness factor, the structure of the data set provided an opportunity to discover whether the behavior of the algorithms appeared to be related to the due-date range.

The parametric values in the data were $R=0.20$ and 0.95. In individual test problems, however, the actual due-date range usually differed from these values. For $R=0.20$, the actual due-date ranges varied from 0.08 to 0.21, while for $R=0.95$ the actual values varied from 0.41 to 0.93. There was thus no overlap between the two parametric settings for R. However, the two settings of the tardiness factor did exhibit some overlap. For $t=0.2$ the actual tardiness factors varied from 0.22 to 0.58, while for $t=0.6$ the actual tardiness factors varied from 0.41 to 0.68. Deviations of actual from parametric values were due to sampling error and to interactions between factors in the dependent cases. Nevertheless, the actual values were satisfactory for detecting effects of the various factors on algorithm performance.

3.4 Dependence

As indicated above, the test data involved job sets in which due-dates were independent of processing times and others in which individual due-dates were dependent on corresponding processing times.

Although this aspect of job data has seldom been explored in reported studies,* it is certainly possible that there is an intrinsic difference between independent and dependent cases. Moreover, one would also expect that more "realistic" tardiness problems are characterized by a certain amount of dependence. While there are many ways of incorporating dependence in the data, the mechanism represented by (5) was a simple way of preserving most of the tardiness factor and due-date range structure of the independent case. The purpose of constructing dependent data sets was to investigate possible systematic differences in algorithm behavior arising from the dependence of due-dates on processing times.

For a given problem size, there were three factors to be investigated (tardiness factor, due-date range, and dependence) yielding eight different parametric cases. Two job sets were created for each of the eight cases, so that 16 job sets were constructed for each value of n or 64 sets in all.

4. EXPERIMENTAL RESULTS

The comparative study was carried out largely as a group project, one which attempted to address the usual problems encountered in comparative investigations [5]. Each of several small groups worked with a particular algorithm in depth and developed a program used in testing the algorithm. Because each group represented a fair amount of programming experience, this arrangement made it unlikely that a particularly inefficient code would be developed for any of the programs. Secondly, the dynamic programming algorithm was tested in both FORTRAN and PL/1, to explore the potential effects of the language used. The results indicated that the programming language itself need not be a critical factor, and all algorithms were coded in FORTRAN. Finally, all run times were observed on the same system: the University of Michigan Terminal System (MTS), which uses an IBM 360/67.

The major experimental results involve the mean solution times observed in the 64 test problems. These are displayed in Table 1. Broadly speaking, the results indicate that the Srinivasan algorithm is the most efficient of the optimal procedures and that the Wilkerson-Irwin algorithm is the fastest of the algorithms tested.

The solution times for dynamic programming exhibit the exponential growth anticipated in this kind of combinatorial problem. The relative success of the other algorithms in improving on this performance reflects their ability to avoid this sensitivity to problem size. For problem sizes of up to eight jobs, there is relatively little difference among the algorithms. In this range, the Srinivasan algorithm is rapid enough so that there would be no justification for using the suboptimal Wilkerson-Irwin algorithm. A brief investigation was made of the individual benefits of Srinivasan's propositions. This experimentation indicated that with the use of only Propositions 1 and 4 the average solution time for the eight-job problems could be reduced to 0.06 sec, which is on a par with the Wilkerson-Irwin method. At the other end of the spectrum, for problem sizes of 15, dynamic programming and branch and bound were an order of magnitude slower than the Emmons and Srinivasan algorithms, which in turn were an order of magnitude slower than the Wilkerson-Irwin algorithm. Because these relationships were consistent for all four problem sizes tested, similar algorithm behavior would be anticipated with larger job sets. The Srinivasan procedure is considerably insulated from the dramatic exponential growth which characterizes dynamic programming; yet, by comparison, the Wilkerson-Irwin procedure's run time grows quite slowly (and in linear fashion) with problem size. For problem sizes at or above 12 jobs, a realistic question is whether to use the faster heuristic, which does not guarantee

*An exception is [9].

optimality, in preference to the slower optimizing technique. This question is examined more fully below.

TABLE 1. *CPU Times (in seconds) Required by Each of the Algorithms on the Test Problems of the Comparative Study*

Problem Size	Optimization Algorithms				Heuristic	
	Dynamic Programming	Branch and Bound	Emmons' Algorithm	Srinivasan's Algorithm	Wilkerson-Irwin	
					Phase 1	Both Phases
8	0.24	0.22	0.12	0.12	0.07	0.07
10	1.21	0.61	0.53	0.29	0.08	0.08
12	5.74	6.06	0.92	0.43	0.09	0.10
15	52.65	[a]63.95	3.15	1.13	0.11	0.12

[a] Two 15-job problems were terminated after 200 seconds without obtaining an optimum.

4.1 Branch and Bound

The solution times for the branch and bound algorithm were the most variable in the study. For some problems the algorithm obtained a solution quite rapidly (some of the 15-job problems were solved in less than 1 sec), but at the other extreme, the maximum solution times were consistently larger than with dynamic programming. Two 15-job runs were terminated after 200 sec of CPU time without obtaining an optimal solution. A closer look at the configuration of the results indicated that the excessive solution times at all problem sizes were frequently associated with large tardiness factors. (This behavior should be expected because the lower bounds are inefficient when several unsequenced jobs have to be tardy.) There was no indication that either of the other factors had an important effect on run times. A more detailed look at the performance of the branch and bound algorithm is provided by Table 2.

In general terms, it appears that there are two types of inefficiencies inherent in the branch and bound method. Most critically, the calculations represented by (1) and (2) are not highly effective mechanisms for obtaining good lower bounds. This means that the procedure is likely to examine very

TABLE 2. *Additional Data on Branch and Bound Solutions.*

Problem Size	CPU time (median)	CPU time (mean)	CPU time (maximum)	Maximum Number Nodes Created	Maximum List Size
8	0.17	0.22	0.63	304	207
10	0.54	0.61	1.85	705	422
12	3.59	6.06	36.17	3999	1478
15	32.56	62.95 [a]	200.00 [a]	6493 [a]	5550 [a]

[a] Two 15-job problems were terminated after 200 seconds without obtaining an optimum.

many partial sequences explicitly and will not substantially curtail their enumeration. To explore the differences between bounds (1) and (2), the eight-job problems were solved twice, once with lower bounds calculated from (1), and then with lower bounds calculated from (2). The latter computation is more time-consuming, of course, but it curtails enumeration much more effectively, as shown in Table 3. In terms of both solution speed and storage requirements, the bound in (2) is preferable, and was used for compiling the data in Table 1.

TABLE 3. *A comparison of Branch and Bound Performance with the use of Bounds Calculated from Eq. (1) and from Eq. (2) (These results were compiled from the sixteen 8-job problems).*

Lower Bound Calculation	CPU time (median)	CPU time (mean)	CPU time (maximum)	Maximum Number Nodes Created	Maximum List Size
Bound (1)	0.23	0.41	1.45	524	298
Bound (2)	0.17	0.22	0.63	304	207

Still, the main problem with a lower bound obtained from (2) is that it ignores available information about all but one of the unsequenced jobs. A modification along the lines of Shwimer's approach [8] could provide a slight remedy, but the nodes created early in the algorithm would still tend to have inefficient bounds. Furthermore, this type of inefficiency would again be more pronounced when the tardiness factor is high, since the potential tardiness of several jobs would often be ignored.

A second inefficiency occurs because the task of managing the list of active subsets can result in considerable overhead. There are alternative ways of managing the list, each involving a different balance of computing requirements and storage requirements. The jumptracking strategy employed in this study would tend to favor computing time at the expense of storage, at least for the configuration of MTS, but it does not seem that a different strategy could achieve very substantial relative improvements in run times under other configurations.

Another approach to solving the problem by branch and bound is to employ a backtracking scheme preceded by a heuristic first phase in order to obtain a near-optimal feasible solution before branching. Then the branch and bound strategy with backtracking—used as a second phase—might be relatively efficient in locating an optimum. (Indeed, the Wilkerson-Irwin procedure might be an excellent candidate for the first phase of such an algorithm.) Therefore, while the results for the pure version of branch and bound were not remarkably good, the approach still holds some promise for tardiness problems. In addition, the efficient design of a complete branch and bound algorithm (i.e., efficient bound calculations, list management, and, perhaps, initial feasible solution) is a topic for future research.

4.2 The Emmons and Srinivasan Algorithms

The reduction phases of the Emmons and Srinivasan algorithms are quite comparable, since they rely on essentially the same properties. The inclusion of Theorem 3 in Emmons' approach can be expected to achieve better reduction than Srinivasan's approach, since it uses only Theorems 1 and 2. Yet the marginal value of this improvement appears to be slight, as indicated by the observations in Table 4, which compares the average reduction obtained by the two methods at each problem size. Once the problem is reduced to a smaller size, the Emmons approach involves a branch-and-enumerate strategy while the Srinivasan approach involves a reduced dynamic programming strategy.

TABLE 4. *Comparison of the Extent to Which the Srinivasan and Emmons Algorithms Reduce the Size of the Original Problem (Each Entry is an Average Proportion of Jobs Sequenced During the Reduction Phases)*

Problem Size	Reduction Capability (%)	
	Srinivasan	Emmons
8	64.1	65.6
10	43.1	43.8
12	52.1	52.6
15	47.5	47.5

In testing the Emmons algorithm we found that the run times were very low as long as no branching was required. At each problem size, however, the maximum observed CPU time was roughly three times as large for the Emmons algorithm as the maximum for the Srinivasan algorithm. The reduced dynamic programming approach is therefore less sensitive to cases in which the reduction phase has only a small effect. This characteristic seems to account for the fact that, overall, the branch-and-enumerate strategy is less efficient than the reduced dynamic programming strategy. As Emmons observed, a natural extension would be to convert the branch-and-enumerate strategy to a branch-and-bound strategy. Associated with this type of extension would be the question of whether a different branching strategy would be more appropriate, but these are both topics for extensive examination in their own right.

The Srinivasan algorithm proved to be the fastest of the optimum-producing techniques, as shown by the results in Table 1. In his original work, Srinivasan observed that median solution times increased only linearly with problem size, and the results of the present study confirmed this finding, as depicted in Figure 1. Nevertheless, it should be remembered that the last phase of the algorithm is basically controlled enumeration and will sometimes resemble the dynamic programming mechanism on which it is based. Thus Figure 1 suggests that the mean solution time still has an exponential tendency, although not nearly as severe as that of pure dynamic programming.

Solution times for the Srinivasan algorithm did not appear to be closely related to tardiness factors, due-date ranges, or the independence property; at least there was no such relation amenable to statistical justification. On the other hand, there was clear evidence that the Emmons algorithm involved excessive computation effort when the due-date range was small. (Maximum CPU times occurred on different 12- and 15-job problems under the Emmons algorithm than under the Srinivasan algorithm.) In both algorithms the run time was directly affected by the number of jobs scheduled by the end of Phase 2. This suggests that while Propositions 1–4 curtail the dynamic programming calculations, they cannot be expected to totally compensate for the effects of problem size in combinatorial situations.

The earlier discussion raised the question of whether all four propositions should be included in Phase 3. Our detailed study of the 8-job problems indicated that Propositions 2 and 3 were counterproductive. In other words, it was quicker to perform certain calculations unconditionally than always to check whether they could be avoided. A more extensive look at this question might make an interest-

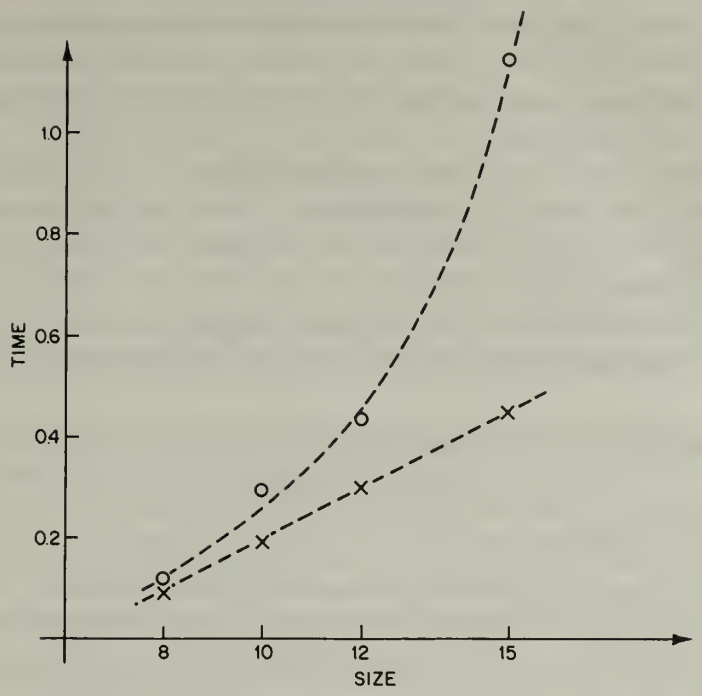


FIGURE 1. A plot of mean CPU times (o) and median CPU times (x) for the Srinivasan algorithm

ing study, although we would anticipate that the four propositions are all beneficial at some large enough problem size. Another fruitful area for research would be the development of additional propositions which would reduce the dynamic programming calculations even further.

4.3 The Wilkerson-Irwin Algorithm

To address the questions raised in section 2.5 about the Wilkerson-Irwin algorithm it is helpful to elaborate on the data in Table 1. As shown in Table 5, Phase 1 of the algorithm frequently produced optima, but was not likely to recognize the solution as optimal. While Phase 1 produced an optimum in about two-thirds of the test problems, the sufficient condition was satisfied in less than one-fourth of these cases. In their original study, Wilkerson and Irwin concluded that the efficiency of their algorithm depended on the due-date range, and our study confirmed this property. All nine problems in which the sufficient condition was satisfied were problems with the larger due-date range. Moreover, all of those were test problems with tardiness factors less than 0.43. Also, Phase 1 produced optima

TABLE 5. *Optimizing Properties of the Wilkerson-Irwin Algorithm*

Problem Size	Number of Problems	Optimum Found	Optimum Recognized	Phase 2 Improved
8	16	14	2	1
10	16	9	1	0
12	16	10	3	2
15	16	9	3	2

in 28 of the 32 problems with the larger range, and failed to produce an optimum only when the tardiness factor was above 0.53. Therefore, Phase 1 is relatively less effective with high tardiness factors as well as with small due-date ranges. The dependence of due-dates on processing times did not appear to be important.

In the 22 problems where Phase 1 was suboptimal, Phase 2 of the algorithm was able to make improvements in only 5, and found an optimum in only 1. Even when improvements were made, they tended to be small. In view of these results, we have concluded that Phase 2 need not be considered an essential part of the procedure.

The great speed of the Wilkerson-Irwin algorithm is obvious from Table 1. To describe the trade-off between solution value and speed, a measure of the algorithm's efficiency was calculated, and is shown in Table 6.

TABLE 6. *Average Efficiency (E) Observed for the Wilkerson-Irwin Algorithm and for the Four Simple Heuristics*

Problem Size	Wilkerson-Irwin		Heuristics			
	Phase 1	Both Phases	SPT	EDD	MRM	MIN
8	0.4	0.3	30.3	8.1	7.8	5.2
10	2.5	2.5	32.5	11.9	11.0	7.2
12	1.9	1.9	37.5	10.5	18.0	5.4
15	1.8	1.7	41.3	11.8	11.3	6.3

The metric used for each problem set was

$$E=\frac{T-T^*}{T}, \tag{7}$$

where E = efficiency of the heuristic solution

T = tardiness value obtained by the heuristic algorithm

T^* = optimal tardiness value.

Thus E represents the potential improvement by an optimal algorithm, expressed as a percentage of the nonoptimal solution. Average values of E over all job sets of a given size are given in Table 6. As shown, the Wilkerson-Irwin algorithm yielded solutions which, on the average, could be improved by 1.6 percent. (The largest efficiency value observed was 16 percent.)

In order to provide perspective, the efficiency statistic was also compiled for other heuristic procedures:

1. Shortest Processing Time (SPT)—Sequence the jobs in nondecreasing order of p_j. This rule is optimal if all jobs must be late.

2. Earliest Due-Date (EDD)—Sequence the jobs in nondecreasing order of d_j. This rule is optimal if all jobs can be completed on time.

3. Montagne's Ratio Method (MRM)—Sequence the jobs in nondecreasing of $p_j \Big/ \left(\sum_{i=1}^{n} p_i - d_j \right)$.

This is a specialized version of a heuristic which Montagne [7] found to be very effective for the weighted version of the tardiness problem.

4. Minimal Heuristic (MIN)—Find a sequence using SPT, EDD, and MRM. Choose the one with minimal tardiness.

The efficiency statistics for these heuristic procedures are shown in Table 6, for comparison with the Wilkerson-Irwin values. Even though the MIN heuristic is computationally very fast, its average efficiency does not approach that of the Wilkerson-Irwin algorithm.

5. CONCLUSION

The principal conclusion in this study is that of the optimizing algorithms tested, the hybrid approach of Srinivasan exhibited the fastest average solution time. For large problems, however, the Wilkerson-Irwin heuristic is substantially faster and provides solutions which are on the average very close to optimal.

The behavior of the algorithms suggests that tardiness factor and due-date range affect algorithm performance. In particular, when a favorable combination of these factors is present, the algorithms can be expected to be more efficient than would otherwise be the case. Such considerations should be recognized in further testing of these algorithms or in comparative studies of new procedures.

The study identified a number of questions worthy of closer examination. Topics for further study in this area include an efficient design of a branch and bound procedure, an improvement of Emmons' branch-and-enumerate strategy, and a cost/benefit analysis of Srinivasan's propositions.

ACKNOWLEDGEMENT

The comparative study reported here began as a class project. Each of the 20 students in the class contributed to various parts of the project, and particular thanks are due to Mr. Gerard A. Dawson and Mr. Charles A. Khuen, III, of the University of Michigan, for their special efforts.

REFERENCES

[1] Conway, R. W., W. L. Maxwell, and L. W. Miller, *Theory of Scheduling* (Addison-Wesley Publishing Co., Reading, Mass., 1967).

[2] Elmaghraby, S. E., "The One Machine Sequencing Problem with Delay Costs," J. Industrial Engineering *19*, 105–108 (Feb. 1968).

[3] Emmons, H., "One Machine Sequencing to Minimize Certain Functions of Job Tardiness," Operations Research *17*, 701–715 (July 1969).

[4] Held, M. and R. M. Karp, "A Dynamic Programming Approach to Sequencing Problems," J. SIAM *10*, 196–210 (Mar. 1962).

[5] Ignizio, J. P., "On the Establishment of Standards for Comparing Algorithm Performance," *Interfaces* (Bulletin of TIMS), Vol. 2, No. 1 (Nov. 1971).

[6] Lawler, E. L., "On Scheduling Problems with Deferral Costs," Management Science *9*, 280–288 (July 1963).

[7] Montagne, E. R., Jr., "Sequencing with Time Delay Costs," Arizona State University Industrial Engineering Research Bulletin No. 5 (Jan. 1969), pp. 20–31.

[8] Shwimer, J., "On the n-Job, One Machine, Sequence-Independent Scheduling Problem with Tardiness Penalties: A Branch-Bound Solution," Management Science *18*, 301–313 (Feb. 1972).

[9] Srinivasan, V., "A Hybrid Algorithm for the One Machine Sequencing Problem to Minimize Total Tardiness," Nav. Res. Log. Quart. *18*, 317–327 (Sept. 1971).

[10] Wilkerson, L. J. and J. D. Irwin, "An Improved Method for Scheduling Independent Tasks," AIIE Transactions *3*, 239–245 (Sept. 1971).

A PROBLEM OF RESTRICTED PARTITIONS

V. R. R. Uppuluri

and

J.A. Carpenter

Mathematics Division
*Oak Ridge National Laboratory**

ABSTRACT

This paper presents a simple algorithm for finding the number of restricted k-partitions of a natural number n. The unrestricted k-partitions of n are expressed as the sum of these restricted k-partitions, called inadmissible, and the admissible k-partitions. The simplicity of the algorithm is striking, though all the implications are unclear.

1. INTRODUCTION

In 1969, Alter and Lientz [1] considered the following combinatorial problem, attributed to Smirnov: Given n objects of $s+1$ categories, in how many ways can the n objects be arranged in a chain so that adjacent objects belong to separate categories? Further, if we have r_i objects in the ith category so that $r_1+r_2+\ldots+r_{s+1}=n$, let $M^{(s+1)}(r_1, r_2, \ldots, r_{s+1})$ denote the number of ways in which these n objects can be arranged in a chain so that adjacent objects belong to separate categories. Based on the recursion

$$(1) \qquad M^{(s+1)}(r_1, r_2, \ldots, r_{s+1}) = \sum_{l=1}^{s+1} \sum_{j=1} (-1)^{j+1} M^{(s+1)}(r_1, \ldots, r_{l-1}, r_{l-j}, r_{l+1}, \ldots, r_{s+1})$$

with

$$M^{(s+1)}(r_1, \ldots, r_{l-1}, 0, r_{l+1}, \ldots, r_{s+1}) = M^{(s)}(r_1, \ldots, r_{l-1}, r_{l+1}, \ldots, r_{s+1}$$

Alter and Lientz [1] tabulated $M^{(3)}(r_1, r_2, r_3)$ (i.e., $s=2$) for the admissible partitions (r_1, r_2, r_3) of $n=3(1)20$. For instance, for $n=7$, the only admissible tripartitions are (4, 2, 1), (3, 3, 1), and (3, 2, 2), and the tripartition (5, 1, 1) gives the value $M^{(3)}(r_1, 1, 1)=0$. Let us call the tripartition (5, 1, 1) inadmissible, and the rest of the tripartitions (into three nonempty parts) admissible. For a given n, it is of interest to find the cardinality of the set of admissible tripartitions (which give rise to nonzero values of $M^{(3)}(r_1, r_2, r_3)$) and generalize the results for $s=3, 4, \ldots$. In this paper, we give a simple algorithm for finding the number $\bar{\alpha}(n, k)$ of inadmissible k-partitions of an integer n. From this, one can find the number $\alpha(n, k)$ of admissible k-partitions of n. We also show the relationship with the tables of partitions.

2. RESTRICTED PARTITIONS

DEFINITION: We define a k-partition $(r_1, r_2, \ldots, r_k)$ of n $(k \geq 2)$, such that $r_1 \geq r_2 \geq \ldots \geq r_k$ (≥ 1) to be *admissible* whenever

*Operated by Union Carbide Corporation for the U.S. Atomic Energy Commission.

$$r_1+r_2+\ldots+r_k=n \tag{2}$$

and

$$r_2+\ldots+r_k \geq r_1-1.$$

Let $\alpha(n, k)$ denote the cardinality of the set of admissible k-partitions of n, and $P(n, k)$ denote the number of partitions of n into exactly k parts. From the definition, we have

$$\alpha(n,k) \leq P(n,k) \qquad \text{for all } n \text{ and } k. \tag{3}$$

Let $\bar{\alpha}(n, k)$ denote the cardinality of the set of inadmissible k-partitions of n. Then we have

$$\bar{\alpha}(n,k)+\alpha(n,k)=P(n,k). \tag{4}$$

We will now prove a lemma, which gives the relationship between inadmissible partitions and exact partitions.

LEMMA:

$$\bar{\alpha}(n,k)=\sum_{j=(k-1)}^{[n/2]-1} P(j,k-1), \tag{5}$$

where $[n/2]$ denotes the greatest integer in $n/2$.

PROOF: The set of all inadmissible k-partitions of n is identical with the set of all k-partitions for which $r_1 > (n+1)/2$. For a given $r_1 (> (n+1)/2)$, we obtain the number of all the inadmissible partitions as $P(n-r_1, k-1)$. The range of r_1 is given by $\left[\frac{n+1}{2}\right]+1, \left[\frac{n+1}{2}\right]+2, \ldots, n-k+1$. Thus we have

$$\bar{\alpha}(n,k)=\sum_{r=\left[\frac{n+1}{2}\right]+1}^{n-k+1} P(n-r,k-1)$$

$$=\sum_{j=k-1}^{[n/2]+1} P(j,k-1).$$

COROLLARY:

$$\bar{\alpha}(2r,k)=\bar{\alpha}(2r+1,k). \tag{6}$$

This follows immediately from the representation given by (5).

3. CUMULATIVE RESTRICTED PARTITIONS

In this section we find the relations between cumulative unrestricted partitions $p(n, k)$, cumulative admissible partitions $a(n, k)$, and cumulative inadmissible partitions $\bar{a}(n, k)$ given by

$$p(n,k)=\sum_{j=1}^{k} P(n,j);\ a(n,k)=\sum_{j=1}^{k} \alpha(n,j)$$

and

$$\bar{a}(n,k)=\sum_{j=1}^{k}\bar{\alpha}(n,j). \tag{7}$$

From (4) and (6), we immediately have

$$a(n,k)+\bar{a}(n,k)=p(n,k). \tag{8}$$

From (6), we have

$$\bar{a}(2r,k)=\bar{a}(2r+1,k). \tag{9}$$

A short table of $\bar{a}(2r,k)$, for $r, k=1, 2, \ldots, 10$ is given in Table 1.

TABLE 1. *Cumulative Inadmissible Partitions* $\bar{a}(2r, k)$

2r \ k	1	2	3	4	5	6	7	8	9	10
2	1	1	1	1	1	1	1	1	1	1
4	1	2	2	2	2	2	2	2	2	2
6	1	3	4	4	4	4	4	4	4	4
8	1	4	6	7	7	7	7	7	7	7
10	1	5	9	11	12	12	12	12	12	12
12	1	6	12	16	(18)	19	19	19	19	19
14	1	7	16	23	27	29	30	30	30	30
16	1	8	20	31	38	42	44	45	45	45
18	1	9	25	41	53	[60]	64	66	67	67
20	1	10	30	(53)	(71)	[83]	90	94	96	97

From Table 1, we see that the numbers $\bar{a}(2r, k)$ can be computed using the following recursion

$$\bar{a}(2r,k)=\bar{a}(2r,1)+\bar{a}(2r-2,2)+\bar{a}(2r-4,3)+\ldots+\bar{a}(2(r-k+1),k),\ r\leq k \tag{10}$$

$$\bar{a}(2r,r)=\bar{a}(2r,r+j),\qquad j=1,2,\ldots$$

with the boundary conditions

$$\bar{a}(2r,1)=1,\qquad \text{for } r=1,2,\ldots$$

$$\bar{a}(1,j)=1,\qquad \textit{for } j=1,2,\ldots.$$

REMARKS: The recursion given by (10) shows that $\bar{a}(n, k)$ can be generated independently of the tables of $p(n, k)$. Further, the relationship

$$\bar{a}(2r+2,k)-\bar{a}(2r,k)=p(r,k-1) \tag{11}$$

shows that $p(n, k)$, the number of unrestricted partitions into at most k parts, can be obtained from the tables of $\bar{a}(2r, j)$. For example, from Table 1, we see that

$$\bar{a}(20,6)-\bar{a}(18,6)=83-60=23,$$

which is equal to $p(9, 5)$. It is not clear, how efficient this procedure of generating $p(n, k)$ may be compared to that of Gupta et al. [2].

From Table 1, we also observe that

$$\bar{a}(2r, k)=\bar{a}(2r, k-1)+\bar{a}(2(r-k+1), k), \tag{12}$$

and deduce

$$\bar{\alpha}(2r, k)=\bar{a}(2(r-k+1), k). \tag{13}$$

For example,

$$71=\bar{a}(20, 5)=\bar{a}(20, 4)+\bar{a}(12, 5)=53+18.$$

From (13) and (6) we note that the values of $\bar{\alpha}(n, k)$ can also be obtained from the tables of $\bar{a}(n, k)$. Alternatively, $\bar{\alpha}(n, k)$ can also be computed by using the relation

$$\bar{\alpha}(n, k)=\bar{a}(n, k+1)-\bar{a}(n, k). \tag{14}$$

From (11) and (8) it follows that

$$a(n, k)=\bar{a}(2n+2, k+1)-\bar{a}(2n, k+1)-\bar{a}(n, k). \tag{15}$$

Thus the values of the cumulative admissible partitions $a(n, k)$ can also be obtained from the tables of $\bar{a}(n, k)$. From (15), the number of admissible partitions $\alpha(n, k)$ can be shown to be equal to

$$\alpha(n, k)=\bar{\alpha}(2n+2, k+1)-\bar{\alpha}(2n, k+1)-\bar{\alpha}(n, k), \tag{16}$$

which can be expressed in terms of $\bar{a}(\cdot, \cdot)$ using (13).

4. SUMMARY

In summary, we note that the tables of $\bar{a}(n, k)$ can be computed easily using the algorithm based on (10). From this table, values of $\bar{\alpha}(n, k)$, $a(n, k)$, $\alpha(n, k)$, $p(n, k)$, and $P(n, k)$ can be easily obtained. Short excerpts of these values are given in Tables 2, 3, 4, and 5.

TABLE 2. *Cumulative Inadmissible Partitions* $\bar{a}(n, k)$

n \ k	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1
4	1	2	2	2	2	2	2	2	2	2	2	2
5	1	2	2	2	2	2	2	2	2	2	2	2
6	1	3	4	4	4	4	4	4	4	4	4	4
7	1	3	4	4	4	4	4	4	4	4	4	4
8	1	4	6	7	7	7	7	7	7	7	7	7
9	1	4	6	7	7	7	7	7	7	7	7	7
10	1	5	9	11	12	12	12	12	12	12	12	12
11	1	5	9	11	12	12	12	12	12	12	12	12
12	1	6	12	16	18	19	19	19	19	19	19	19

TABLE 3. *Inadmissible Partitions* $\bar{\alpha}(n, k)$

n \ k	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
4	1	1	0	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0	0	0
6	1	2	1	0	0	0	0	0	0	0	0	0
7	1	2	1	0	0	0	0	0	0	0	0	0
8	1	3	2	1	0	0	0	0	0	0	0	0
9	1	3	2	1	0	0	0	0	0	0	0	0
10	1	4	4	2	1	0	0	0	0	0	0	0
11	1	4	4	2	1	0	0	0	0	0	0	0
12	1	5	6	4	2	1	0	0	0	0	0	0

TABLE 4. *Cumulative Admissible Partitions* $a(n, k)$

n \ k	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	1	1	1	1	1	1	1	1	1	1
3	0	1	2	2	2	2	2	2	2	2	2	2
4	0	1	2	3	3	3	3	3	3	3	3	3
5	0	1	3	4	5	5	5	5	5	5	5	5
6	0	1	3	5	6	7	7	7	7	7	7	7
7	0	1	4	7	9	10	11	11	11	11	11	11
8	0	1	4	8	11	13	14	15	15	15	15	15
9	0	1	6	11	16	19	21	22	23	23	23	23
10	0	1	5	12	18	23	26	28	29	30	30	30
11	0	1	7	16	25	32	37	40	42	43	44	44
12	0	1	7	18	29	39	46	51	54	56	57	58

TABLE 5. *Admissible Partitions* $\alpha(n, k)$

n \ k	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0
3	0	1	1	0	0	0	0	0	0	0	0	0
4	0	1	1	1	0	0	0	0	0	0	0	0
5	0	1	2	1	1	0	0	0	0	0	0	0
6	0	1	2	2	1	1	0	0	0	0	0	0
7	0	1	3	3	2	1	1	0	0	0	0	0
8	0	1	3	4	3	2	1	1	0	0	0	0
9	0	1	5	5	5	3	2	1	1	0	0	0
10	0	1	4	7	6	5	3	2	1	1	0	0
11	0	1	6	9	9	7	5	3	2	1	1	0
12	0	1	6	11	11	10	7	5	3	2	1	1

REFERENCES

[1] Alter, R. and B. P. Lientz, "Applications of a Generalized Combinatorial Problem of Smirnov," Nav. Res. Log. Quart., *16*, 543–547 (1969).

[2] Gupta, H., C. E. Gwyther, and J. C. P. Miller, *Tables of Partitions*, Cambridge University Press, 1958.

A NOTE ON SOME SERIES REPRESENTATIONS OF THE INTEGRAL OF A BIVARIATE NORMAL DISTRIBUTION OVER AN OFFSET CIRCLE*

Dennis C. Gilliland

Michigan State University

and

Eldon R. Hansen

Lockheed Research Laboratory

ABSTRACT

Consider the problem of computing an offset circle probability under a normal distribution. One approach is to utilize an infinite series representation in which case it is important to have rapid convergence and a good upper bound on the error introduced by consideration of only a finite number of terms of the series. We relate three seemingly different series representations. In particular we show how two series representations for the bivariate case can be obtained by specializing more general results of Harold Ruben.

1. INTRODUCTION

Ruben [11] has developed infinite series representations for the probability content of a hypersphere under a multivariate normal distribution and has shown that these are mixture representations, in the sense of Robbins and Pitman [10], for certain values of a scale parameter p appearing within his expansions. The purpose of this note is two fold. First we observe a small technical point concerning conditions under which mixtures are realized. Second we use this result in the important bivariate case to show that the Ruben series is a mixture when p is taken to be the harmonic mean of the two variances and that the series is then the same as that due to Gilliland [3]. Ruben has observed that when p is taken to be the smallest of the variances his series is a mixture. We show that in the bivariate case with this choice of p, the Ruben series is the same as that Groenewoud, et al. [5] have used to compute a table of offset circle probabilities. That a representation is a mixture is important for bounding the series truncation error. Several examples are given to illustrate computation with the series and to compare rapidity of convergence for the two specifications of p.

2. THE SERIES

Since part of the purpose of this note is to compare three published series representations all of which use different notation, it was decided that this note employ notation from the paper containing the most general results. For this reason we will follow the notation of Ruben [11] and write his representation (3.1) as

$$H_{n,\underset{\sim}{A},\underset{\sim}{b}}(t) = \sum_{j=0}^{\infty} c_j(p) F_{n+2j}(t/p), \qquad \text{for all } t \geq 0. \tag{1}$$

*This work was supported in part by NSF Grants GP-1459 and GP-23480.

Here $H_{n;\underset{\sim}{A},\underset{\sim}{b}}(t)$ represents the probability content of an origin centered hypersphere of radius $t^{1/2}$ under the multivariate normal distribution which has mean $(a_1^{1/2}b_1, \ldots, a_n^{1/2}b_n)$ and diagonal covariance matrix $\underset{\sim}{A}$ with diagonal elements $a_1 \leq a_2 \leq \ldots \leq a_n$. Equivalently, $H_{n;\underset{\sim}{A},\underset{\sim}{b}}(t)$ is the cumulative distribution function of the nonhomogeneous quadratic form $\Sigma_1^n a_i(X_i - b_i)^2$, where $X_1, \ldots, X_n$ are independent univariate standard normal random variables. In (1) F_{n+2j} denotes the cumulative distribution function for the chi-square distribution with $n+2j$ degrees of freedom. The expansion is a mixture representation in the sense of [10] for values of p which make the coefficients $c_j(p)$ a probability distribution on the nonnegative integers j. Let C denote the set of p values for which (1) is a mixture representation.

REMARK 1: In (1) if $c_j(p) \geq 0$ for all j, then $\Sigma_0^\infty c_j(p) = 1$.

PROOF: The probabilistic interpretation of the left hand side of (1) yields $1 \geq \Sigma_0^\infty c_j(p)F_{n+2j}(t/p)$ for all t. If $c_j(p) \geq 0$ for all j, then the monotone convergence theorem yields $1 \geq \lim_{t\uparrow\infty} \Sigma_0^\infty c_j(p)F_{n+2j}(t/p) = \Sigma_0^\infty c_j(p) \lim_{t\uparrow\infty} F_{n+2j}(t/p) = \Sigma_0^\infty c_j(p)$. The reverse inequality follows from letting $t \uparrow \infty$ in $H_{n;\underset{\sim}{A},\underset{\sim}{b}}(t) \leq \Sigma_0^\infty c_j(p)$.

The above remark shows that treatment of the condition $\Sigma_0^\infty c_j(p) = 1$ is not necessary for determining C. It suffices to treat the nonnegativity alone; therefore, a simplification of the analysis [11, pp. 565–566] follows. Remark 1 is applicable to any representation $G(x) = \Sigma_0^\infty c_j G_j(x)$, where G, G_0, G_1, . . . are probability cumulative distribution functions. In particular it applies to (4.8) of [11] and the corresponding coefficients $d_j(p)$.

REMARK 2: Let p, n, $\underset{\sim}{A}$ and $\underset{\sim}{b}$ be fixed. If

$$H_{n;\underset{\sim}{A},\underset{\sim}{b}}(t) = \sum_{j=0}^{\infty} c_j^* F_{n+2j}(t/p), \quad t \geq 0,$$

where the $c_j^* \geq 0$, then $c_j^* = c_j(p)$, $j = 0, 1, \ldots$.

PROOF: The c_j^* give a mixture representation by the proof of Remark 1. Therefore, $c_j^* \leq 1$, $j = 0, 1, \ldots$ and the series $\Sigma c_j^* F'_{n+2j}(t/p)$ is seen to be uniformly convergent in t on any interval $T_1 \leq t \leq T_2$, where $0 < T_1 < T_2 < \infty$. Hence, $pH'_{n;\underset{\sim}{A},\underset{\sim}{b}}(t) = \Sigma c_j^* F'_{n+2j}(t/p)$, $t > 0$. The bound (4.14) of [11] show that the series $\Sigma c_j(p)F'_{n+2j}(t/p)$ converges uniformly in t on any interval $T_1 \leq t \leq T_2$, where $0 < T_1 < T_2 < \infty$, and, therefore, $pH'_{n;\underset{\sim}{A},\underset{\sim}{b}}(t) = \Sigma c_j(p)F'_{n+2j}(t/p)$, $t > 0$. By equating the series and appealing to the uniqueness of coefficients in a power series expansion, we get the result $c_j^* = c_j(p)$, $j = 0, 1, \ldots$.

The uniqueness across mixtures is known as the identifiability of the mixing distribution. Teicher [13] has results which establish the identifiability of mixtures $\Sigma c_j^* F_j$ of Gamma distributions F_j.

REMARK 3: If $n = 2$ and $p^{-1} = \frac{1}{2}(a_1^{-1} + a_2^{-1})$,, that is, $p = a_h$, the harmonic mean of the variances a_1 and a_2, then (1) is a mixture and the same as the representation (12) of [3]. If $n = 2$ and $p = a_1$, then (1) is a mixture and the same series as that developed in [5].

PROOF: We rewrite (12) of [3] in the notation of Ruben [11] and with the convention $a_1 \leq a_2$. For this purpose we note that $\beta = (a_1 + a_2)/4a_1 = \frac{1}{2}a_2/a_h$, $z^2 = t/a_2$, and $P_m(\beta z^2) = P_m(t/2a_h) = F_{2+2m}(t/a_h)$ so that (12) of [3] can be written as

$$H_{2;\underset{\sim}{A},\underset{\sim}{b}}(t) = \sum_{j=0}^{\infty} B_j F_{2+2j}(t/a_h). \tag{2}$$

The nonnegativity property $B_j \geq 0$ follows from the representation for the coefficients derived by

Hansen [8], which, in the present notation reads $B_j = a_h(a_1a_2)^{-1/2}$ exp $\{-\frac{1}{2}(b_1^2+b_2^2)\}$ $[(a_2-a_1)/(a_1+a_2)]^j|H_j(z_0)|^2/j!$ where H_j is the Hermite polynomial (2.7) of [11] and z_0 is the complex number, $z_0 = [a_1a_2/(a_2-a_1)]^{1/2}$ $(-b_1a_1^{-1/2} + ib_2a_2^{-1/2})$. (In case $a_1 = a_2$, the coefficients B_j are obtained as limiting values. For a detailed derivation of this representation of the B_j see [4].) By Remark 1 it follows that (2) is a mixture and by Remark 2, $B_j = c_j(a_h), j = 0, 1, \ldots$. We now consider the series representation appearing on page x of [5] with the convention $a_1 \leq a_2$ and note that $a^2 = t/2a_1$ so that $S_m(a^2) = P_{m-1}(t/2a_1) = F_{2m}(t/a_1)$. Thus, the series derived by Groenewoud, et al. [5] can be written

$$H_{2;\underset{\sim}{A},\underset{\sim}{b}}(t) = \sum_{j=0}^{\infty} D_j F_{2+2j}(t/a_1). \tag{3}$$

The nonnegativity of D_j is evident from the representation for D_j given in [5, p. x] so that by Remark 1 (3) is a mixture. By Remark 2 it follows that $D_j = c_j(a_1), j = 0, 1, \ldots$. (That $c_j(a_1)$ provides a mixture is implied for the general n case by (5.8) of [11].)

Ruben [11, p. 565] has observed that with $p = a_h$ and $\underset{\sim}{b} = \underset{\sim}{0}$, (1) is a mixture representation if the a_i^{-1} are symmetrically located about a_h^{-1}. For $n = 2$, the symmetry requirement is automatically satisfied and Remark 3 extends Ruben's observation to arbitrary $\underset{\sim}{b}$ so that the offset circle case is covered. This is important because of the interest in computing offset circle probabilities. With mixture representations a simple bound on the truncation error obtains (cf. (9) of [10], (5.13) of [11] and (21) of [3]). Also, the analysis [11, p. 567] indicates a rapid rate of convergence for the series (1) in case $p = a_h$ and $\underset{\sim}{b}$ is small. One can expect the series (2) to be more efficient than series (3) for computing offset circle probabilities for small offset distances.

In case $a_1 = a_2$ the representations (2) and (3) reduce to (27) of [10] specialized to the bivariate case and when $b_1 = b_2 = 0$, (2) reduces to a series due to Esperti [2, (14)]. As noted earlier the series (3) has been used to compute a table of $H_{2;\underset{\sim}{A},\underset{\sim}{b}}(t)$ [5]. Numerical techniques have also been used to compute tables of the probability $H_{2;\underset{\sim}{A},\underset{\sim}{b}}(t)$. Some of these tables are referenced in the review papers [1], [7]. If only one or two place accuracy is required either interpolation across existing tables or a quick approximation discussed by Grubbs [6] is preferable to computation with (1). Before turning to examples of offset circle computations with (2) and (3) we note some other series representations that have been developed; for example, see [9] and, for the n-dimensional case, [12]. However, these last mentioned series do not appear well-adapted for computation.

3. EXAMPLES

We will now compute using the series (2) and (3).

EXAMPLE 1: Consider Problem 1 of [3] where $a_1 = 1$, $a_2 = 4$, $b_1 = 0.2$, $b_2 = 0.2$, and $t = 4$. Using the recursion relationship (3.5) of [11] we compute $c_0(a_h) = 0.8e^{-0.04}$, $c_1(a_h) = 0.032e^{-0.04}$, $c_2(a_h) = 0.13312e^{-0.04}$, and $c_3(a_h) = 0.0168277333 \ldots e^{-0.04}$. The particular values $F_j(t/a_h) = F_j$ (2.5) are not readily available in tables, but are easily computed using the Poisson probability representation for a chi-square tail probability. The results to the fifth decimal place are $F_2(2.5) = 0.71350$, $F_4(2.5) = 0.35536$, $F_6(2.5) = 0.13153$, $F_8(2.5) = 0.03827$, and $F_{10}(2.5) = 0.00913$. Using the first four terms of (2) we have that

$$H_{2;\underset{\sim}{A},\underset{\sim}{b}}(4) \doteq e^{-0.04}[0.5708 + 0.0114 + 0.0175 + 0.0006] = 0.5768.$$

Since (2) is a mixture, the first four terms underestimate $H_{2;\underset{\sim}{A},\underset{\sim}{b}}(4)$ by less than $F_{10}(2.5)\left[1 - \sum_0^3 c_j\right] =$

0.0005 (confer (5.13), [11]) and the probability 0.577 is correct to the three displayed decimals. In contrast, the first four terms of series (3) give

$$H_{2;\underset{\sim}{A},\underset{\sim}{b}}(4) \doteq e^{-0.04}\,[0.4323 + 0.1188 + 0.0363 + 0.0101] = 0.5741.$$

The series (3) is seen to converge more slowly than the series (2) in this example. The truncation error bound by Groenewoud et al. [5, p. xiii] does not apply to such a small number of terms of this particular series in contrast to (5.13) of [11]. Interpolation recommended for use in table [5] gives the result 0.559 and the approximations [6] (see Example 1 of [6]) yield the estimates 0.570, 0.573, 0.558, 0.569, and 0.544.

EXAMPLE 2: If we remove the offset in Example 1, that is, take $b_1 = 0$, $b_2 = 0$, we can expect (2) to converge very rapidly. The first three nonzero terms of series (2) yield

$$H_{2;\underset{\sim}{A},\underset{\sim}{0}}(4) \doteq [0.5708 + 0.0189 + 0.0004] = 0.5901.$$

The bound on the truncation error is $F_{12}(2.5)[1 - (c_0 + c_2 + c_4)] = 0.0000$ so that the probability 0.590 is correct to the number of displayed digits. In contrast, the first three nonzero terms of series (3) yield

$$H_{2;\underset{\sim}{A},\underset{\sim}{0}}(4) \doteq [0.4323 + 0.1114 + 0.0341] = 0.5778.$$

Table [2] gives the result 0.590095 and Table [5] gives 0.59010.

EXAMPLE 3: Consider a rather large offset $b_1 = 1$, $b_2 = 2$ along with $a_1 = 1$, $a_2 = 4$, and $t = 4$. In this case series (2) yields

$$H_{2;\underset{\sim}{A},\underset{\sim}{b}}(4) = e^{-2.5}[0.5708 + 0.4549 + 0.1536 + 0.0474] = 0.1007.$$

The bound on the underestimate is $F_{10}(2.5)\left[1 - \sum_0^3 c_j\right] = 0.0058$ so that to three places the exact probability is between 0.101 and 0.107. The series (3) yields

$$H_{2;\underset{\sim}{A},\underset{\sim}{b}}(4) = e^{-2.5}[0.4323 + 0.4084 + 0.2362 + 0.1067] = 0.0972.$$

Interpolation across two entries in Table [5] gives the result 0.1018.

4. CONCLUSIONS

The series (3) derived by Groenewoud et al. to develop a table of offset circle probabilities [5] can be obtained by specializing the series (1) derived by Ruben [11] with $p = a_1$, $n = 2$. The series (2) derived by Gilliland [3] can be obtained by specializing the series (1) with $p = a_h$, $n = 2$. Since series (2) and (3) are mixtures, there exist simple upper bounds on their truncation errors. Either series can be programmed or used with a desk calculator to compute offset circle probabilities to any specified accuracy. Analysis of Ruben [11, p. 567] and some examples indicate that series (2) will converge faster than series (3), at least for small offset distances.

REFERENCES

[1] Eckler, A. R., "A Survey of Coverage Problems Associated with Point and Area Targets," Technometrics **11**, 561–589 (1969).

[2] Esperti, R. V., "Tables of the Elliptical Normal Probability Function," Defense Systems Division, General Motors Corp., Detroit (1960).

[3] Gilliland, D. C., "Integral of the Bivariate Normal Distribution over an Offset Circle " J. Am. Stat. Assoc. **57**, 758–768 (1962).

[4] Gilliland, D. C., and Hansen, E. R., "On a Series Representation of the Integral of the Bivariate Normal Distribution over an Offset Circle," RM–247, Department of Statistics and Probability, Michigan State University (1970).

[5] Groenewoud, C., Hoaglin, D. C., and Vitalis, J. A., "Bivariate Normal Offset Circle Probability Tables." CAL No. XM–2464–G–1, Cornell Aeronautical Laboratory, Inc., Buffalo, N.Y. (1967).

[6] Grubbs, F. E., "Approximate Circular and Noncircular Offset Probabilities of Hitting," Operations Research **12**, 51–62 (1964).

[7] Guenther, W. C. and Terrango, P. J., "A Review of the Literature on a Class of Coverage Problems," Annals of Mathematical Statistics **35**, 232–260 (1964).

[8] Hansen, E. R., "Problems and Solutions," SIAM Review **6**, 461–462 (1964).

[9] Laurent, A. G., "Bombing Problems—A Statistical Approach," Operations Research **5**, 75–89 (1957).

[10] Robbins, H. E. and Pitman, E. J. G., "Application of the Method of Mixtures to Quadratic Forms in Normal Variates," Annals of Mathematical Statistics **20**, 552–560 (1949).

[11] Ruben, H., "Probability Content of Regions under Spherical Normal Distributions IV: The Distribution of Homogeneous and Nonhomogeneous Quadratic Functions of Normal Variables. Annals of Mathematical Statistics **33**, 542–570 (1962).

[12] Shah, B. K. and Khatri, C. G., Distribution of a Definite Quadratic Form for Non-Central Normal Variates," Annals of Mathematical Statistics **32**, 883–887 (1961).

[13] Teicher, H., "Identifiability of Mixtures," Annals of Mathematical Statistics **32**, 244–248. (1961).

☆U.S. GOVERNMENT PRINTING OFFICE: 1974-541-389/3

INFORMATION FOR CONTRIBUTORS

The NAVAL RESEARCH LOGISTICS QUARTERLY is devoted to the dissemination of scientific information in logistics and will publish research and expository papers, including those in certain areas of mathematics, statistics, and economics, relevant to the over-all effort to improve the efficiency and effectiveness of logistics operations.

Manuscripts and other items for publication should be sent to The Managing Editor, NAVAL RESEARCH LOGISTICS QUARTERLY, Office of Naval Research, Arlington, Va. 22217. Each manuscript which is considered to be suitable material for the QUARTERLY is sent to one or more referees.

Manuscripts submitted for publication should be typewritten, double-spaced, and the author should retain a copy. Refereeing may be expedited if an extra copy of the manuscript is submitted with the original.

A short abstract (not over 400 words) should accompany each manuscript. This will appear at the head of the published paper in the QUARTERLY.

There is no authorization for compensation to authors for papers which have been accepted for publication. Authors will receive 250 reprints of their published papers.

Readers are invited to submit to the Managing Editor items of general interest in the field of logistics, for possible publication in the NEWS AND MEMORANDA or NOTES sections of the QUARTERLY.

NAVAL RESEARCH LOGISTICS QUARTERLY

MARCH 1974
VOL. 21, NO. 1
NAVSO P-1278

CONTENTS

OFFICE OF NAVAL RESEARCH
Arlington, Va. 22217